MULTI-DIMENSIONAL WAVE FRONT SENSING ALGORITHMS FOR EMBEDDED TRACKING AND ADAPTIVE OPTICS APPLICATIONS

THESIS

Christopher C. Wood, First Lieutenant, USAF
AFIT/GE/ENG/06-57

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/06-57

MULTI-DIMENSIONAL WAVE FRONT SENSING ALGORITHMS FOR EMBEDDED TRACKING AND ADAPTIVE OPTICS APPLICATIONS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Christopher C. Wood, BS

First Lieutenant, USAF

March 2006

AFIT/GE/ENG/06-57

MULTI-DIMENSIONAL WAVE FRONT SENSING ALGORITHMS FOR EMBEDDED TRACKING AND ADAPTIVE OPTICS APPLICATIONS

Christopher C. Wood, BS
First Lieutenant, USAF

Approved:

/signed/	
Dr. Stephen C. Cain (Chairman)	date
/signed/	
Dr. Richard K. Martin (Member)	date
/signed/	
Dr. Yong C. Kim (Member)	date

AFIT/GE/ENG/06-57

Abstract

Current tracking and adaptive optics techniques cannot compensate for fast-moving extended objects, which is important for ground-based telescopes providing space situational awareness. To fill this need, a vector-projection maximum-likelihood wave-front sensing algorithm development and testing follows for this application. A derivation and simplification of the Cramer-Rao Lower Bound for wave-front sensing using a laser guide star bounds the performance of these systems and guides implementation of a vastly optimized maximum-likelihood search algorithm. A complete analysis of the bias, mean square error, and variance of the algorithm demonstrates exceptional performance of the new sensor. A proof of concept implementation shows feasibility of deployment in modern adaptive optics systems. The vector-projection maximum-likelihood sensor satisfies the need for tracking and wave-front sensing of extended objects using current adaptive optics hardware designs.

One More Roll

We toast our faithful comrades now fallen from the sky

And gently caught by God's own hand to be with him on high.

To dwell among the soaring clouds they knew so well before

From dawn patrol and victory roll at heaven's very door.

And as we fly among them there we're sure to hear their plea

"Take care, my friend, watch your six, and do one more roll... just for me."

Gerald (Jerry) Coffee, Captain, USN (Ret.)

Hanoi, 1968

Acknowledgments

I would like to thank my Dad for his unending support without whom I would not be here today, and Michelle for her care and understanding as this thesis consumed the time and attention that she deserved. This work would not have been possible without the guidance provided by my advisor, Dr. Stephen Cain, or the education provided by the committee members Dr. Richard Martin and Dr. Yong Kim.

Christopher C. Wood

Table of Contents

Page

Page

Page

Page

List of Figures

List of Tables

MULTI-DIMENSIONAL WAVE FRONT SENSING ALGORITHMS FOR EMBEDDED TRACKING AND ADAPTIVE OPTICS APPLICATIONS

I. Introduction

Atmospheric turbulence affects clarity of anything in space viewed through large telescopes. Machines that perform optical tracking of moving targets or provide high-resolution imaging must correct for turbulence effects by detecting the distorted wave-front caused by turbulence to prevent loss of tracking ability or image corruption [16]. The capability to detect distortion in the wave-front, or relative position changes of an image, is often embedded or built into the wave-front sensor and processing algorithms as a part of the adaptive optics system [16]. Modern adaptive optics systems allow for wave-front correction with only guide stars and small, extended sources, sometimes requiring post-processing of gathered data [12]. A new maximum-likelihood wave-front sensing algorithm embedded in proven adaptive optics designs could enhance detection for non-ideal conditions and real-time operations [5]. What and where atmospheric turbulence is, how adaptive optics attempt to overcome the effects of this turbulence, and why these optics systems need improvement all become clear in the following sections.

1.1. Background and Motivation

1.1.1. The Effects of Atmospheric Turbulence

Many factors on earth, such as natural processes and terrain features, affect weather significantly; however, the main driver of turbulence is the sun’s uneven heating of the earth's surface. The uneven heating causes convection currents and wind spawning circular currents, eddies, which trap varying temperatures throughout the atmosphere causing variations of the index of refraction thereby distorting the wave-front. Figure 1 shows the first two major layers, the troposphere and stratosphere, containing 99.9 % of the earth's atmosphere, and whose turbulence is responsible for the majority of light distortions [13]. The figure also indicates an average temperature gradient; a few realistic sample temperature gradients as seen through different columns of air; and other sources of turbulence such as shearing winds, terrain, and natural processes feeding convection. The results of these sources of turbulence can combine to distort a wave-front as it passes through different temperature gradients in the earth's atmosphere.

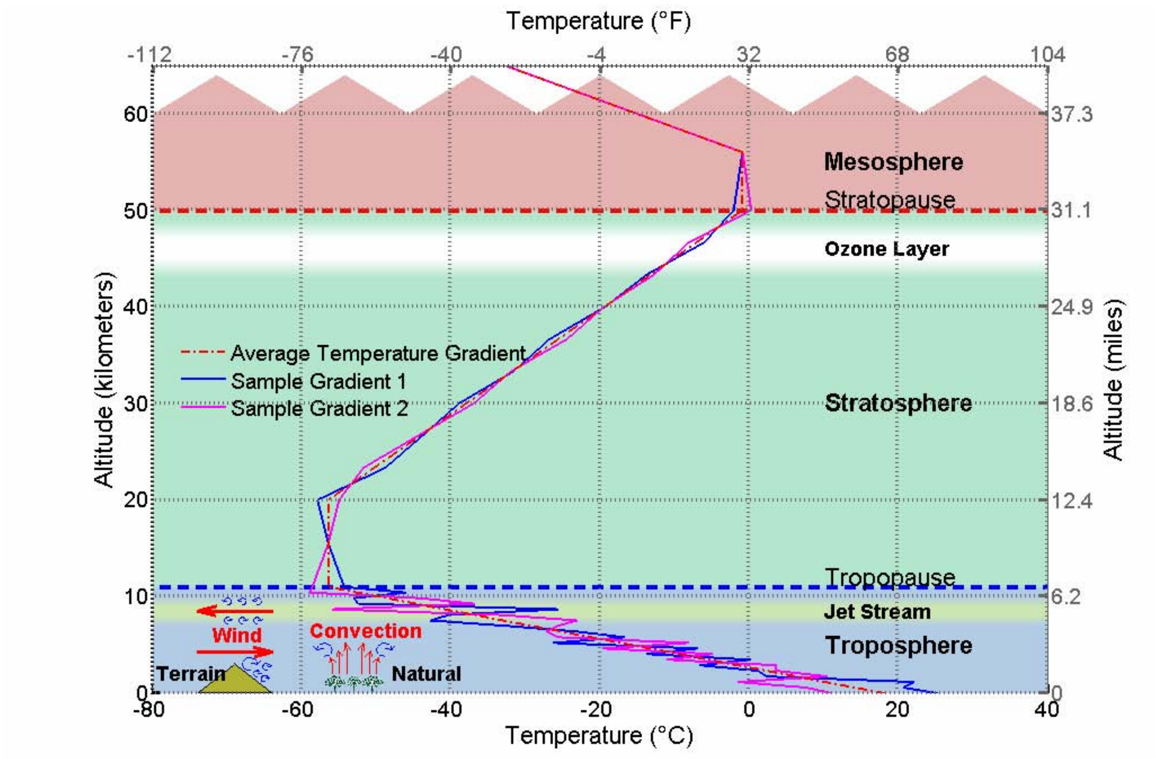


Figure 1. Temperature Gradients and Turbulence Sources in the Atmosphere [13]

A clearer view of how a wave-front distorts and how the wave-front initially forms is available in Figure 2. A point source, or a distant star, emits light, which travels outward from the star much as ripples travel outward from a pebble thrown in a pond. When these "waves" are far away from the source, they appear as a straight line, forming a wave-front. Researchers often model the propagating waves as a two-dimensional Fourier Transform. Much like taking the Fourier Transform of a single point in time results in a straight line in the frequency domain, a point source in space transforms to a plane wave related to spatial frequency rather than temporal frequency [8]. The wave-front does not distort much as it passes through the stratosphere, as temperature variations seldom occur there; however, the troposphere severely distorts the wave-front due to the numerous opportunities for eddies to form and trap temperature variations. The result is a corrupted wave-front that, when focused onto an imaging device produces a blurry and distorted image bearing little resemblance to the original object.

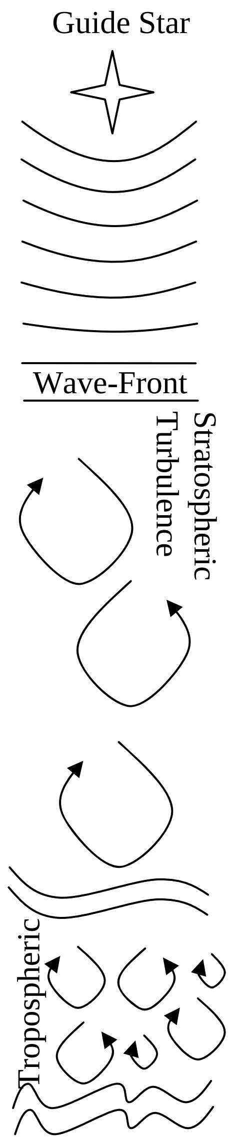


Figure 2. Distorting of Wave-Front Moving Through Atmosphere [12]

In addition to using phase screens to model the temperature variations and the relative refractive index changes directly at different altitudes, researchers use Zernike polynomials, or "Zernikes", to characterize the distortions in the wave-front itself [16]. As opposed to a rectangular based set of polynomials, the Zernike polynomials describe a set of circular-based, two-dimensional functions corresponding to the circular opening in a telescope or other imaging device [16, 22]. These models are crucial to correcting the wave-front in an Adaptive Optics (AO) system.

1.1.2. Adaptive Optics Solutions

Although there are many applications for adaptive optics in modern imaging systems, the basic structure as shown in Figure 3 for a general large telescope system remains relatively constant across the applications [16].

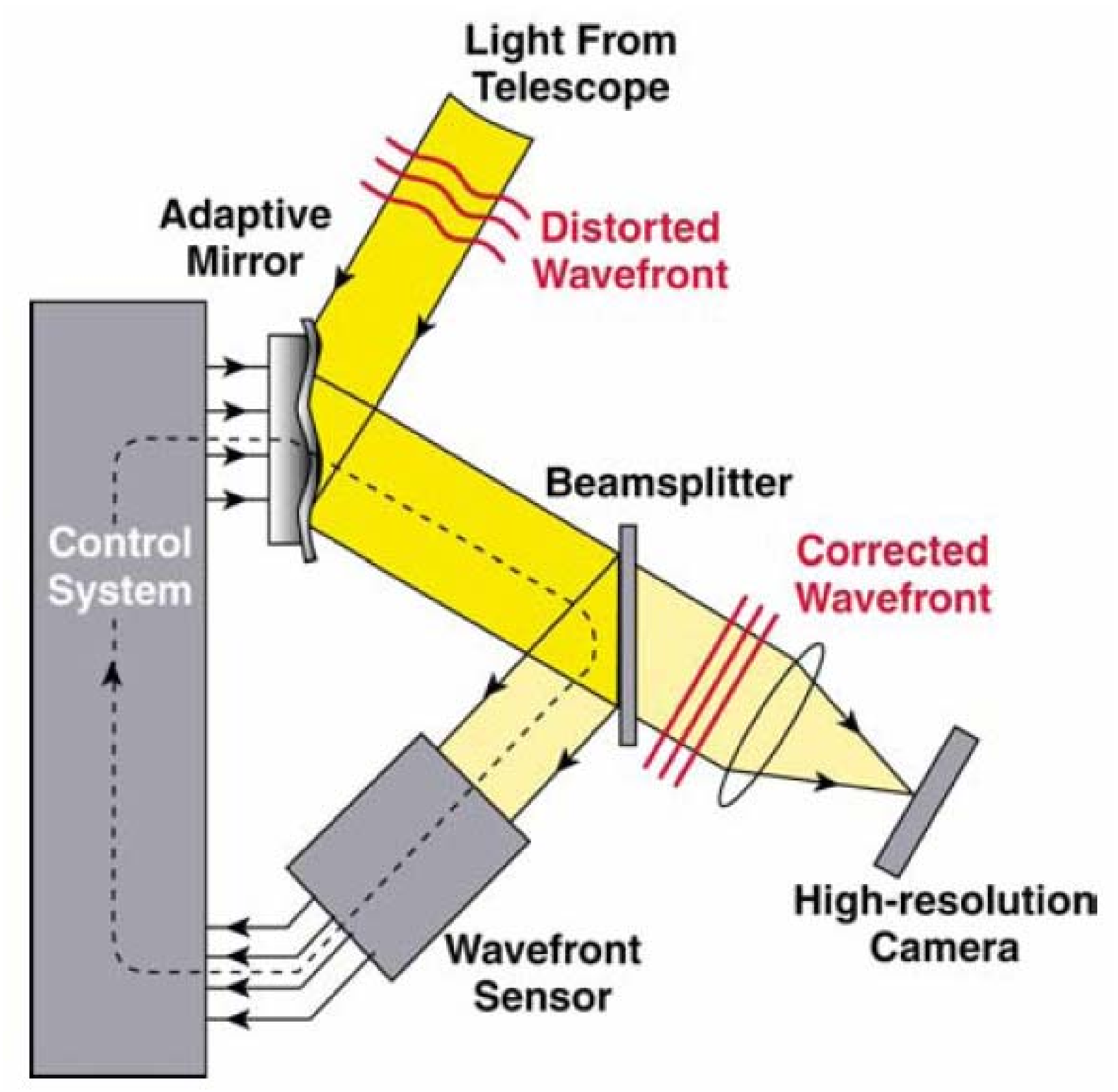


Figure 3. Typical Adaptive Optics System Based on a Large Telescope [21]

A simple trace through the system reveals that light enters through the telescope lens with a distorted wave-front, and then reflects from an adaptive mirror, a mirror that can deform using mechanical actuators, which is initially flat as there is no information to correct the wave-front. The light then continues to a fifty-fifty beam-splitter sending half of the light into a lens, which focuses the light onto a high-resolution imaging device, and the other half to the wave-front sensor. The first portion of the wave-front sensor both optically and electrically detects measurable parameters of the wave-front, passing that information to an algorithmic portion of the wave-front sensor to estimate the parameters for later modeling. Since the wave-front sensor is the heart of this system, this thesis concerns itself with the algorithmic portion of the wave-front sensor. These estimates pass to the reconstructor in the control system, which builds a model of the wave-front and then applies that information to a known model for the adaptive mirror to attempt wave-front correction. If the wave-front has a lag or dip in it causing the light to arrive later than expected, the mirror must have a corresponding bump to accelerate the light back to its appropriate phase to compensate for the distortion. Once an initial estimate corrects the wave-front, additional wave-front sensing refines the current estimates and detects further changes, producing higher quality results for future images.

1.1.3. Problems and Need for Improvement in Wave-Front Sensing

The applications for higher quality imaging span the gamut, from ground-based and space-based telescopes to military applications such as the Airborne Laser and even medical services such as measuring aberrations, or deformities, in an eye. These applications drive the need for better quality imaging and improvements in wave-front sensing.

As with any scientific research, improvements require a metric by which to measure results and draw conclusions. To this end, knowing the structure for a Cramer-Rao lower bound (CRLB) would provide, independent of the estimation technique, an analytical method to judge the efficacy of current and proposed wave-front sensing algorithms. Once known, the Cramer-Rao lower bound can also guide research for improving current estimation techniques as well as developing new estimation approaches to manage more complex imaging scenarios.

A complex situation of interest is imaging of extended objects, or light sources that do not conform to the definition of a point source, such as a satellite in orbit, the surface roughness of the sun, a scud missile, or even a truck on a highway. Tracking a satellite in orbit allows for space situational awareness, or imaging of foreign assets, without placing costly assets in space; however, it requires wave-front updates for this extended object, the satellite, at an incredible rate of 1,000 Hz or greater due to the speed in which the satellite moves. A complication to the satellite-tracking scenario stems from the typical optical tracking system, which causes the image to fill the field of view and allows new information to enter the scene while tracking, defeating current fast-acting sensors. The surface intensity variation, or roughness, of the sun is a unique problem in that the image gathered has extremely low-contrast features for tracking or correlation, disabling most modern wave-front sensors; but imaging of the sun is necessary to predict communication outages and solar weather in general. The scud missile, truck, and other daytime AO applications represent a class of objects whose backgrounds, like the sun, are not black reducing the contrast, and require rapid wave-front updates for the dynamic turbulence between the object and imaging device. Imaging extended objects, although merely a collection of point sources with spatial reference to each other, is not the only type of imaging that current wave-front sensors can have poor performance.

Occasionally, atmospheric turbulence results in a tip-tilt, represented by Zernike polynomials two and three, beyond the physically measurable range of the sensor causing an unknown in the collection of estimated parameters and preventing the reconstructor from modeling the wave-front. This unknown occurs when the tip or tilt is so great that the majority of the image moves off the detector leaving the algorithm a small amount of information to work with. Modern sensors are not capable of controlling such a situation, and the entire adaptive optics system suffers when a single sensor cannot acquire an accurate estimate for the wave-front.

Although the optical and electrical properties of current sensors potentially support the previously mentioned improvements, the algorithms currently in use do not; therefore, an investigation of a vector-projection, maximum-likelihood-correlating wave-front sensor guided by Cramer-Rao lower bounds and simulation experiments will proceed.

1.2. Summary of Current Techniques

Several factors limit the performance of current adaptive optics techniques preventing the ability to track or perform wave-front sensing for fast-moving, extended objects, or low-contrast objects. The largest contributor to these limitations is the wave-front sensor, which provides the necessary information for the adaptive optics system to correct the wave-front deformities.

There are two main categories of wave-front sensors, low-order wave-front sensors and advanced wave-front sensors, both of which are capable of detecting wave-front distortions. The advanced wave-front sensors typically produce better performance through higher order computations and more complex algorithms; however, most cannot image an extended object and none are capable of the tracking application as closed loop speeds are currently very low [6]. Current low-order wave-front sensors provide slightly lower imaging performance, but operate at up to 1000 Hz, allowing for tracking and other fast-moving imaging applications [16]. These simpler estimation techniques include numerous wave-front sensors; however, only the easily implemented and fast-operating Shack-Hartmann and Short-Wavelength Adaptive Techniques (SWAT) wave-front sensors are common today [16]. Both of these sensors use a centroid-based algorithm to estimate tip and tilt, and this algorithm can have extremely poor performance when attempting wave-front sensing or tracking on an extended object [16]. The simplicity of the centroid algorithm suggests that a more complex and statistically based algorithm could surpass these sensors in performance, possibly retaining the operating speed while tracking or performing wave-front sensing on extended objects.

Research indicates the theoretical vector-projection maximum-likelihood wave-front sensor can achieve the performance of a low-order wave-front sensor for tracking and wave-front sensing of guide stars while providing suitable performance for imaging extended objects [5]. This wave-front sensor uses the same hardware system as the SWAT wave-front sensor; however, the algorithm is a maximum-likelihood estimation technique, which provides correlation capability for an extended object while maintaining performance for point sources [5]. Currently only limited simulated statistical characterization of this sensor is available and the tracking application requires a modern processor to implement this more complex algorithm [5].

1.3. Contributions and Scope

It is the goal of this research to quantify the efficacy of a vector-projection, maximum-likelihood-correlating wave-front sensor for tracking extended objects based on a satellite application, as well as a couple wave-front sensing applications, through three facets [5].

The first contribution is generalized model for the Cramer-Rao lower bound with assumptions allowing for future applications provides the analytical basis for research. The CRLB should be applicable to any type of wave-front sensor.

The second contribution is an algorithmic analysis to increase the temporal performance of the new complex maximum-likelihood algorithm to allow simulations that thoroughly characterize the noise-independent bias of the algorithm resulting in a third contribution as well as the noise statistics of mean squared error (MSE) and variance (VAR) for a fourth contribution. The variance directly compares to the Cramer-Rao lower bound to reveal limitations in the algorithm. The search algorithm developed in this phase contributes to other applications requiring a fast and complete algorithm to perform the search of functions with special properties such as maximum-likelihood.

The fifth contribution is a proof of concept for a feasible method of developing this algorithm for embedded hardware implementation and a complete plan for implementation offers insight to researchers in the field looking for feasible solutions. The third criterion is complete when a single working implementation emerges; however, multiple revisions provide further utility.

This three-faceted exploration secures a concrete approach to the research, development, and implementation of a vector-projection, maximum-likelihood-correlating wave-front sensor.

1.4. Approach/Methodology

The three-faceted investigation above, with the provided motivation, is a template that guides the organization of both the research and this document. A thorough investigation of current techniques with appropriate discussion of relevant subjects provides the necessary foundation for research. This leads to development of the tracking and wave-front sensing application environments for producing realistic simulations and allowing accurate characterization of the new algorithm. Theoretical analysis develops the CRLB for wave-front tilt estimates, which provides input for development of a fast, compact, and complete search algorithm for discovering the peak likelihood. From the validated algorithm extends a focused hardware implementation. The results of extensive simulations provide the bias, mean squared error, and variance statistics characterizing the algorithm for tracking and numerous wave-front sensing applications. The research concludes with a synopsis and areas of further research, allowing for future contributions to the body of knowledge regarding tracking and wave-front sensing.

II. Background

2.1. Current Wave-Front Sensing Limitations

Modern imaging of extended objects requires either a stable point source in the field of view or complex optics and algorithms to detect the wave-front correctly across the lens of the telescope. The extended source typically forces researchers, astronomers, and field users to find or create a guide star close to the extended object they wish to view. The applications mentioned previously, particularly imaging large objects or tracking fast moving targets, are difficult or impossible to realize using nearby or artificial guide stars. Wave-front sensing and tracking is possible due to the complex system mentioned in the introduction; however, the key components are the wave-front sensor and the algorithm to determine tip and tilt. The following describes the typical tip-tilt only detectors and a few more complex wave-front detection methods, directly compares and summarizes the features of each sensor, and finally presents areas of potential research given this information.

2.2. Low Order Wave-Front Sensors

2.2.1. Shack-Hartmann Wave-Front Sensor

The most widely employed wave-front sensor uses the Hartmann test to estimate the linear, lower order Zernikes, two and three, and currently has the best overall real-time performance [16]. Both the hardware structure and the algorithm to gather offset, or tip and tilt, information lead to a simple mathematical model stemming from the elementary nature of the sensor as described below [17].

Figure 4 illustrates a typical Shack-Hartmann sub-aperture array in one-dimension and a single sub-aperture in Figure 5 indicates a linearly tilted wave-front and the corresponding offset in two-dimensions when focused [16]. The sub-apertures must be small enough to meet the Nyquist sampling criterion to ensure that the curved wave-front is linear in the region measured by the sensor driving the overall number of sub-apertures [8]. The Nyquist rule applies to any sub-aperture type system, as well as another generalized rule that imposes a requirement of approximately one adaptive optics channel, sub-aperture, per turbulence coherence radius r_0 as a minimum, independent of the telescope size [14]. Larger numbers of sub-apertures implies smaller sizes; however, this larger number of sensors can introduce more noise into the system and decrease light, degrading overall system performance [5].

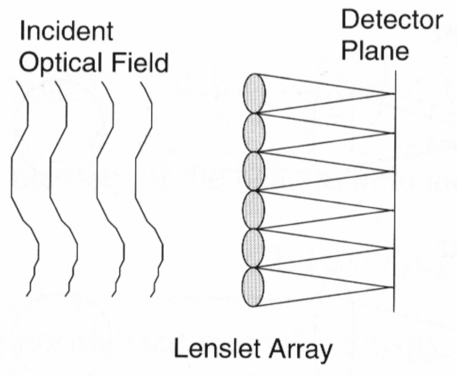


Figure 4. Wave-Front Sensor Array [16]

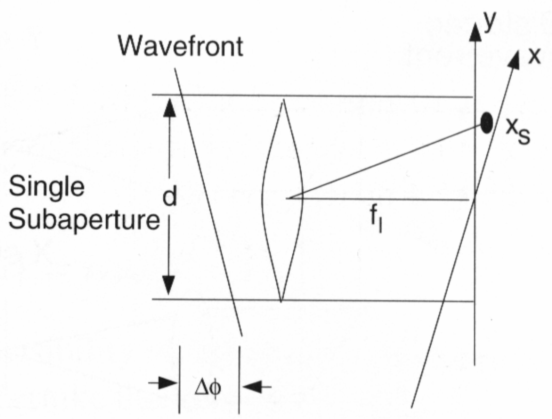


Figure 5. Single Wave-Front Sensor Element [16]

These sub-apertures consist of a lenslet array, which focuses the light onto a charge coupled device (CCD) array for the individual wave-front sensors (WFS) [16]. The CCD readout, where the information collects, is the second opportunity for significant noise injection before the wave-front algorithm begins processing.

The algorithm driving a Shack-Hartmann wave-front sensor is a simple two-dimensional centroiding algorithm [17]. Each intensity readout multiplies a linear position number, then average together, and finally the total power in the image divides the result for the centroid in one-dimension and then repeats for the next dimension. This operation takes a minimal amount of time allowing greater than 100 Hz operation, and provides quality results for guide stars and moderately extended objects [12]. There is a lower bound on error for shot noise, or quantization noise; however, it is somewhat restrictive and only applies to the Shack-Hartmann wave-front sensor and guide stars [16]. The simplistic nature of this algorithm lends itself to improvement in accuracy as time permits such investigations.

One performance improvement for the Shack-Hartmann sensor came from research at MIT Lincoln Laboratory; the short wavelength adaptive techniques wave-front sensor, which splits the incoming light to two lenslet arrays and two CCDs oriented at 90° to each other. The performance improvement stems from allowing the CCD to gather all of the charge in the image into vector readout, or a projection, and then performing a one-dimensional centroiding algorithm for each orientation [2]. Although image projection allows for both faster readout and lower readout noise, it decreases the brightness of the original image, decreasing the signal to noise ratio (SNR) making an accurate estimate less likely.

2.2.2. Shearing Interferometer

A more complex wave-front sensor not typically considered outside of academia that strictly estimates Zernikes two and three is the lateral shearing interferometer [23]. Although the physical implementation of a single shearing interferometer can be simple, the algorithm to retrieve a usable tip-tilt requires a high degree of effort, and the model for the wave-front sensor system clearly indicates the non-mathematical foundation of the apparatus and the amount of processing required to retrieve phase information [16].

The physical apparatus splits the incoming light several times encompassing the entire wave-front of the sensor to perform filtering and polarization for different measurement techniques [16]. Once split, the beam splits again before shearing in orthogonal directions by a tunable amount, only to recombine with the non-sheared version and create an interference pattern as shown in Figure 6 [23]. This pattern has a direction relationship to the wave-front tilt, and a sinusoidal nature over time allowing researchers to correct the wave-front in a reasonable time frame [16].

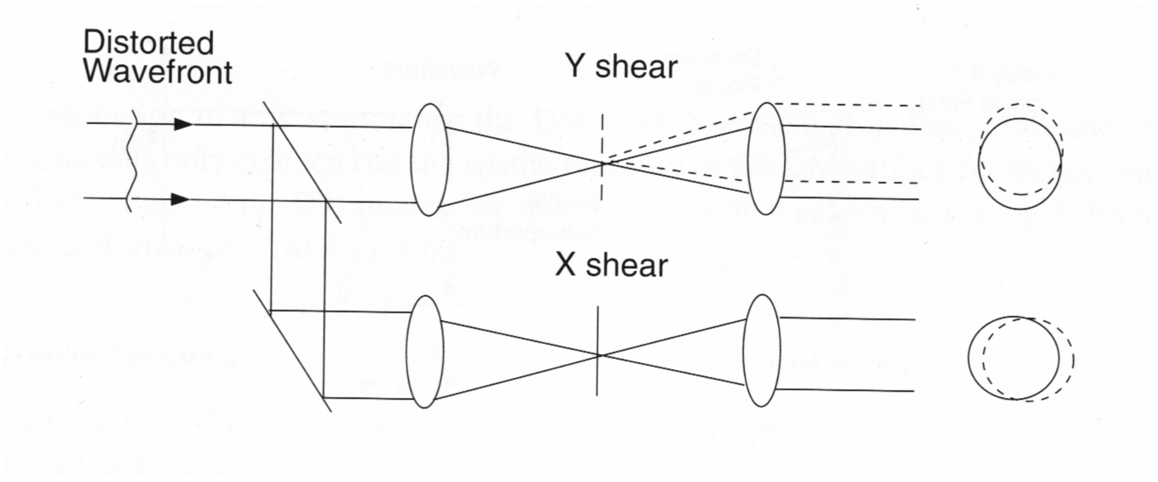


Figure 6. Shearing Interferometer Final Stage Operation [16]

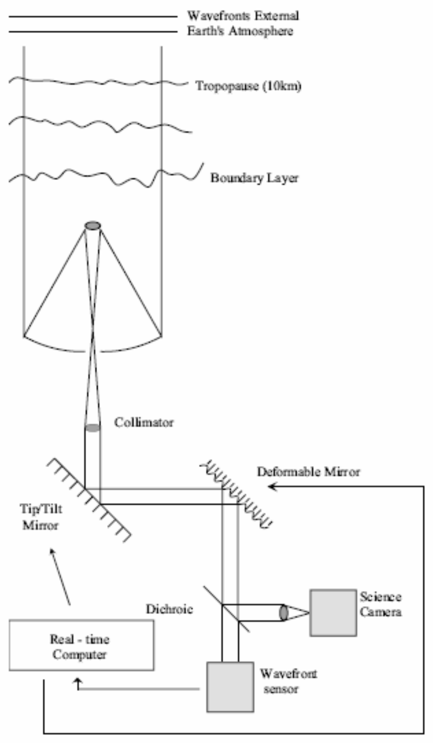
Decoding the phase from this interference pattern takes many forms; however, all algorithms lead to similar results with a modest time delay and correct operation for point sources [16]. The limitation to point sources stems from the expectation of a plane wave at the receiver. Without a point source, the interference pattern includes noise from the shape of the extended object and corrupts the output waveform. The lateral shearing interferometer is the most tunable wave-front sensor, but tuning is crucial to match the Shack-Hartmann sensor under ideal conditions.

2.3. Advanced Wave-Front Sensors

2.3.1. Curvature Wave-Front Sensor

A promising new wave-front sensor is the curvature wave-front sensor. Curvature sensing has additional requirements for the adaptive optics system by adding a secondary deformable mirror [1]. The hardware for this system relies on the Shack-Hartmann or other low-order wave-front sensing detectors; however, the algorithm driving the higher order results, Zernikes four and above, takes the same information and performs a superior analysis at an elevated processing cost [1]. The key for this method is the requirement for an accurate tip-tilt sensor in order to perform correctly, thus requiring the best low-order Zernike sensor/estimator possible.

Aside from the addition of a deformable mirror shown in Figure 7, the first mirror corrects for tip-tilt and the second correct higher order Zernikes, the fundamental concept of sampling the image changes for a curvature wave-front sensor [1]. Sampling of in focus and out of focus images occurs simultaneously at a minimum of 1 kHz rate using a special parabolic mirror as in Figure 8.



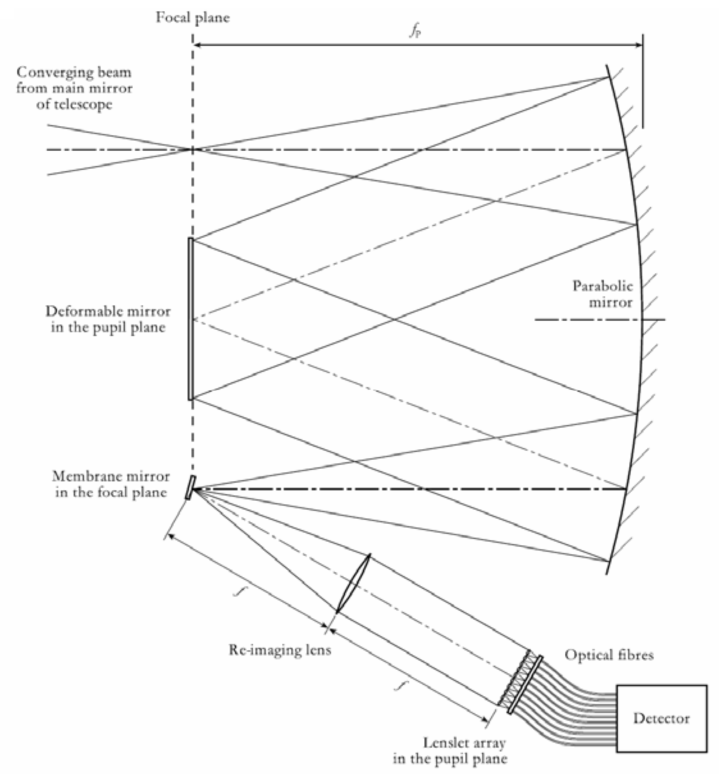

Focal plane
f_P
Converging beam from main mirror of telescope
Parabolic mirror
Deformable mirror in the pupil plane
Membrane mirror in the focal plane
Re-imaging lens
Optical fibres
Detector
Lenslet array in the pupil plane

Figure 7. Curvature Adaptive Optics Setup [1]

Figure 8. Curvature Sensing Setup [1]

The multi-phase sampling allows wave-front correction over the entire visible spectrum, and provides the flexibility to operate at lower frequencies as well; however, like the shearing interferometer it assumes a point source is the subject of the image [1]. The complexity of this system forces the researcher to justify the modest performance gain with the significant hassle required to install, setup, and maintain this system.

2.3.2. Phase Diversity Wave-Front Sensor

Possibly the simplest structure of all wave-front sensors appears in the phase diversity wave-front sensor. This type of sensor concentrates on superior algorithms as it is not capable of the basic autocorrelation algorithms generally used in wave-front reconstruction using the other wave-front sensors [3]. The required maximum-likelihood techniques require tremendous processing power, as addressed on a smaller scale for the theoretical sensor, and typically apply to offline de-convolution of an image rather than real-time correction of wave-front aberrations [6].

A beam splitter and a second imaging device at a greater focal length is all the additional hardware required for this sensor to estimate at least the first 21 Zernike polynomials [3]. Once estimated, the coefficients of the Zernike polynomials allow for de-convolution of the image with the atmosphere, allowing for imaging when guide stars are not available [10]. This process takes an inordinate amount of time, and is not capable of sustaining an adaptive optics system in real-time for fast-moving objects or rapidly changing turbulence; however, enough information is available for post-processing methods. Low contrast scenes are still difficult to image with this method as the SNR decreases significantly.

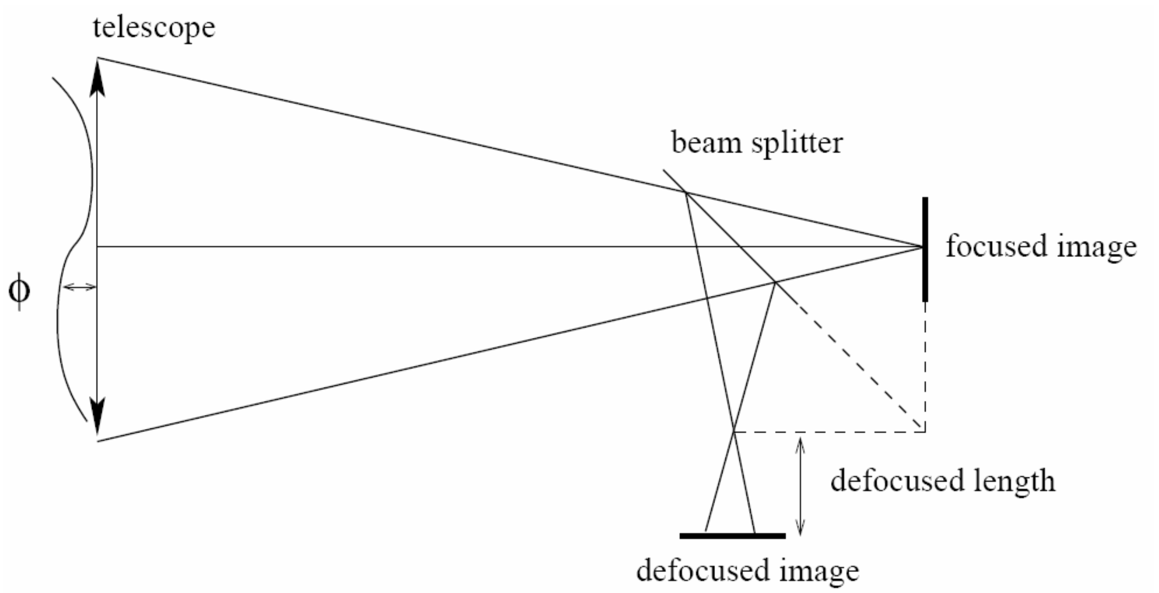


Figure 9. Typical Phase Diversity Hardware Setup [3]

2.4. Theoretical Maximum Likelihood WFS

A vector-projection maximum likelihood wave-front sensor builds upon the design of the Shack-Hartmann and extends the SWAT wave-front sensor requiring no major hardware changes from the SWAT design. This hardware setup provides the same readout noise reduction as the SWAT sensor, while the algorithm used to detect tip and tilt surpasses centroiding in photon noise rejection, particularly for extended objects, at a cost of higher computation time [5].

The hardware portion of this sensor adds an additional beam splitter just before the original Shack-Hartmann sensor exactly as the SWAT wave-front sensor does, with the split beam feeding an identical, but rotated 90°, array of sub-apertures and CCD elements. The CCD structures mirror the SWAT device as well by using vector readouts of the images creating projections of the original image in two-dimensions. The algorithm then uses these projections independently for the autocorrelation related maximum likelihood estimation of tip and tilt [5]. Characterization for the setup and some statistics already exist from a previous work; therefore, extension into the tracking and characterization for wave-front sensing should be simpler [5].

2.5. Comparison and Summary

Limitations in current adaptive optics technologies constrain the ability to perform ad-hoc imaging of fast-moving, extended, or low-contrast objects. These limitations generally stem from the wave-front sensor, as it is the key component in an adaptive optics system. Table 1 shows, using a scale of Excellent-Good-Marginal-Poor, Shack-Hartmann and curvature wave-front sensors have good performance for various object types and a respectable response time, easily allowing for common use today [1, 5, 16, 17]. The lateral shearing interferometer and phase diversity wave-front sensors have other advantages as seen in Table 2 that outweigh the detriments of such complex systems [3, 16, 23]. The theoretical maximum likelihood sensor provides excellent image tracking capabilities while maintaining a low complexity and high response time making it an ideal candidate for further research [5].

Table 1. Performance Comparison for Common Wave-Front Sensors

WFS	Performance on Given Object			Speed	Complexity	
	Point	Extended	Low Contrast		Hardware	Algorithm
Shack-Hartmann	Excellent	Marginal	Assumed Poor	Excellent	Low	Low
Shearing Interferometer	Good	Poor	Poor	Good	Medium	Medium
Curvature	Excellent	Marginal	Poor	Marginal	Medium	High
Phase Diversity	Excellent	Excellent	Good	Poor	Low	Very High
Maximum Likelihood	Theoretically Excellent	Theoretically Good	Assumed Marginal	Good	Low	Medium

Table 2. Additional Known Advantages and Disadvantages of Wave-Front Sensors

WFS	Other Advantages	Other Disadvantages
Shack-Hartmann		Requires Small, High Contrast Object for Good Estimation
Shearing Interferometer	Very Adaptable to Current Environment	Requires Extensive Tuning
Curvature		Requires Tip-Tilt Estimation First for Edges of Wave-Front
Phase Diversity	Allows De-Convolution of Image	Only Offline Operation
Maximum Likelihood	Possible Off-Edge Lock Capability; Multiple SW Realizations Possible	Requires Estimate of the True Image

2.6. Possible Areas of Investigation

Before research beings to attempt a performance improvement, a benchmark for comparison is always a good idea. To this end, a Cramer-Rao lower bound for wave-front sensing should establish a solid baseline. Additional applications for the maximum likelihood wave-front sensor are of interest, to include integration with phase diversity algorithms, near and off-edge performance of guide stars, and multi-spectral maximum likelihood analysis. To make a feasible sensor, the algorithm must be capable of real-time operations within a closed-loop system requiring algorithmic analysis and decomposition. Taking this decomposition of implementable algorithms it should be possible to perform hardware simulations and analysis.

III. Modeling

Investigating new topics requires thorough modeling of the known environment to guide the research and provide adequate testing of results from these analyses and simulations. This chapter of the thesis defines the programming and simulation software, the methods to generate realistic data within these programming environments, and the relevant facts surrounding these modeling techniques. Verification and validation for expected performance of investigation results requires not only the modeling capability and understanding but also development platforms for software and hardware simulations and fabrication.

The majority of software validation and simulation uses MATLAB version 7.0.4.365 (R14) Service Pack 2 with the Signal Processing Toolbox, executing both the simulated environment and sensor model under test. However, some algebraic, differential, and statistical validation uses Mathematica version 5.2 for symbolic manipulation and verification of complex formulas. Simulated hardware verification requires a different development environment and uses Altera's Quartus II version 5.1 Build 176 for both hardware modeling and testbench simulation. These development platforms provide a broad yet firm foundation for design and assessment of image and signal processing technologies through both software and hardware elaboration and simulation capabilities.

The objective in modeling images is to provide the most realistic and best-case scenarios for sensor characterization, while providing the sensors with enough information to exceed modern performance expectations.

3.1. Image Modeling Parameters

As with any modeling, the parameters for modeling are often more important than the modeling itself, and the parameters controlling image creation listed in Table 3 are no exception.

3.1.1. Wavelength

Since atmospherically induced optical tilt bends different wavelengths of light much like a prism, ideally a sensor should receive only one wavelength to perform estimation as an image further distorts when combining different wavelengths. To avoid further distortion, all created images include the assumption that the wavelength is quasi-monochromatic, including a range of 0.05 μm of wavelengths, and fixed both spatially and temporally.

Table 3. Image Modeling Parameters and Simulated Ranges

Parameter	Description	Simulated Range
Wavelength	Wavelength of Light Received	Quasi-Monochromatic and Fixed
Sampling	Nyquist or Higher Sampling Rate	1 to 2 times Nyquist
Image Size	Size in Pixels of Captured Image	8 to 64 Pixels Square
Light Level	Sum Total of Light at Receiver	Guide Star: 100 to 1,000 Photons Extended Object: 6,000-20,000 Photons
Background Intensity	Additive Stray Light in Receiver	0 to 1 Photon per Pixel

While the useful information contained in other wavelengths should produce similar characteristic results, the tilt information from one additional wavelength will further correct wave-front error by characterizing the true path of light through the atmosphere, an effect not modeled or investigated. Only genuine images with real data will define the actual frequency of light used for modeling as sampling requirements for real images require this information.

3.1.2. Sampling

Once the light passes through the atmosphere and enters the telescope, it is necessary to sample the point spread function (PSF) appropriately according to the Nyquist sampling theorem to avoid aliasing of frequency content in the image [16]. Starting from the cutoff frequency of the optical transfer function (OTF) of the lens shown in Equation 1, Nyquist sampling chooses the minimum sampling frequency to be at least twice this cutoff frequency [16].

$$f_c = \frac{D}{\lambda} \tag{1}$$

$$f_s \geq 2 \cdot f_c = 2 \cdot \frac{D}{\lambda} \tag{2}$$

where

f_c = Telescope Optic Cutoff Frequency (radians^{-1})

f_s = Sampling Frequency (radians^{-1})

D = Lens Diameter (meters)

λ = Light Wavelength (meters)

Given that the wavelength of light remains constant for this modeling, any adjustment required for Nyquist sampling will occur either by adjusting the sampling rate or the lens diameter. For an image at a fixed distance from the telescope, Equation 2 combines with Equation 3, which assumes the small angle approximation, and then controls the wavelength and aperture diameter based on the actual angular coverage of a pixel.

$$\alpha = \frac{dx}{z} = \frac{1}{f_s} \tag{3}$$

where

α = Angular Coverage of Pixel (radians)

dx = Size of Pixel on Object (meters)

z = Distance to Object (meters)

Over-sampling has the added benefit of aiding an interpolator for better estimation results, but it also decreases the light available to each pixel causing detrimental effects explained in Section 3.5. Nyquist sampling theory does not address the resolution limits between objects in the image; therefore, the Rayleigh, Dawes, or Sparrow criteria do not contribute to modeling and completely ignored to provide an idealized characterization of the sensors [15]. Since the sampling frequency is twice the cutoff frequency as determined by the diffraction limited effect of a telescope opening, a shift of one Nyquist pixel is analogous to a slope in the frequency domain of π radians, or one-half of one wave of tilt, via the Fourier shift theorem [11].

3.1.3. Image Size

Determining the actual image size requires knowledge of the search area for wave-front tilt as well as the particular requirements of a wave-front sensor, and in an effort to guide the search area, a quick derivation of the statistical nature of tilt follows. As often used in a Monte Carlo simulation when generating a phase screen, or layer of turbulence causing tilt at a set altitude, the Cholesky factorization of the Zernike polynomials' covariance matrix multiplies a vector of zero-mean, unit variance Gaussian random variables to create a set of statistically accurate Zernike coefficients [16]. The tilt therefore remains Gaussian and originates from the low order elements of the covariance matrix, captured in Equation 4 for Zernikes two and three [9].

$$\sigma_{tilt}^2 = 0.448\left(\frac{D}{r_0}\right)^{-\frac{5}{3}}, \left\{radians^2\right\} \tag{4}$$

where

σ_{tilt}^2 = Variance of Tilt (radians^2)

D = Diameter of Aperture (meters)

r_0 = Fried Parameter (meters)

This formula yields variances, and subsequently standard deviations, less than one when compared to a wave of tilt, which is 2π radians, for either a large telescope or a small turbulence coherence radius creating large values of D/r_0, allowing for computation of a search window. An image supporting searches of plus or minus four waves of tilt would provide a worst case of no less than 99.99 percent possible tilt coverage, requiring a search space of plus or minus eight pixels [9]. Prior temporal analysis explains this derivation in further detail and indicates that in a closed loop system, as an adaptive optics system provides, values of tilt beyond one to two standard deviations are exceedingly unlikely unless the adaptive optics system loses lock requiring a greater search space [4].

Different sensors require various image sizes to allow for optimal performance, and although telescopes often include four-by-four pixel images for Shack-Hartmann sensors this image size severely limits the range of detection for tilt measurements, thus the current most complex version of this sensor defines the lower bound of an eight-by-eight pixel image as larger image sizes degrade read-out performance. The theoretical Maximum Likelihood sensor relies on a minimum of twice the number of pixels to correlate with compared to the desired search space for extended objects, and to achieve this sixteen pixel search space, the theoretical sensor requires a minimum size of thirty-two pixels. With the middle ground of image sizes fixed, the upper bound stems from a forward-looking perspective with respect to greater turbulence and superior accuracy. An important assumption as image sizes grow beyond approximately thirty-two by thirty-two pixels is isoplanism of the observed wave-front, which may be true for a natural guide star (NGS), is probably not accurate for a satellite in orbit, and most likely incorrect for a laser guide star (LGS). It is also important to highlight that the parameterization of image size, although somewhat dependent upon sampling, is independent of light-level and background intensities.

3.1.4. Light-Level (Total Intensity)

Another parameterized variable in image creation is the total intensity of the image, or light-level, which the telescope controls based on the object imaged, the amount of light split through the beam splitter to the wave-front sensor, and the integration time of the sensor. To avoid temporal distortions caused by quickly changing turbulence, a short integration time is desirable; and for modeling purposes, all images assume an ideal integration time of 100 μs [16]. Adaptive optics entails obtaining light levels ranging from ten to sixty percent of the total light received by the telescope, with the least amount of light required being the most desirable as the light for wave-front sensing detracts from that available to the primary sensor. To show the performance of all sensors in acceptable light levels and to demonstrate significant trends as light-levels increase or decrease, this modeling uses a modest search range near the lowest light-levels commonly used. It is interesting to note that, as clarified in Section 3.5, when light level decreases the effective signal-to-noise ratio also decreases making a correct estimate of the tilt less likely. This is only one way that the contrast ratio, or ratio between the highest and lowest intensities in the image, changes, modifying the background intensity also changes this hidden parameter.

3.1.5. Background Intensity

The last considered parameter indicates the efficiency of the optical and electrical components of an imaging system, and can significantly affect the results of certain sensor models, which expect a black background to perform estimation. The background light level typically cumulates from stray light in the imaging system as well as stray electrons in the image capture device, causing a lower contrast ratio and subsequently a lower SNR. A perfect imaging system could have a background light intensity of zero, while a very poor system might aggregate an overwhelming background of one photon per pixel or more for the light levels in the parameterization range. This background effectively resides beneath the signal represented by an image, and can cause a smooth function such as a Gaussian to change shape significantly.

3.2. Image Creation

To avoid unknown effects in simulation, image creation requires explicit knowledge of the characteristics of an image with nearly precise knowledge of the statistics expected from turbulence effects on that image allowing for modeling validation and reliable simulations.

3.2.1. Two-Dimensional Gaussian (Simulated Laser Guide Star)

The two-dimensional Gaussian represented by Equation 5 is possibly the simplest image type to model, and genuinely represents the nature of an artificially generated laser guide star, which can allow wave-front correction for extended objects. It is important to note the assumption of independence and equality for the variance in both dimensions of the Gaussian, which may not be correct for true hardware and atmospheric turbulence. Additionally, there is an extra parameter *C* to adjust the light level of the image, which simply scales the complete picture and does not modify the Gaussian in any other respect. Figure 10 and Figure 11 illustrate the true form of this image without noise for a three-dimensional view, projection images in both axes, and a two-dimensional representation.

$$i(x, y) = C\left(2\pi\sigma^2\right)^{-1} e^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}} \tag{5}$$

where

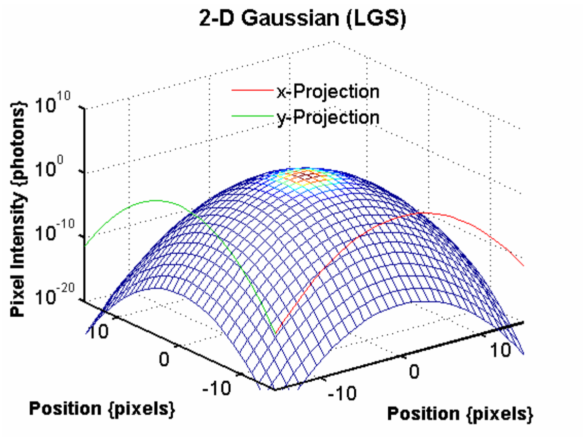
i = Representation of 2-D Gaussian Image (photons)

x, y = Pixel Locations in the Image (pixels, $\in$ Integers)

C = Total Intensity of Image (photons)

σ = Standard Deviation (pixels, $\in$ Positive Reals)

Assuming $\sigma = \sigma_x = \sigma_y$ and the Two Dimensions are Independent



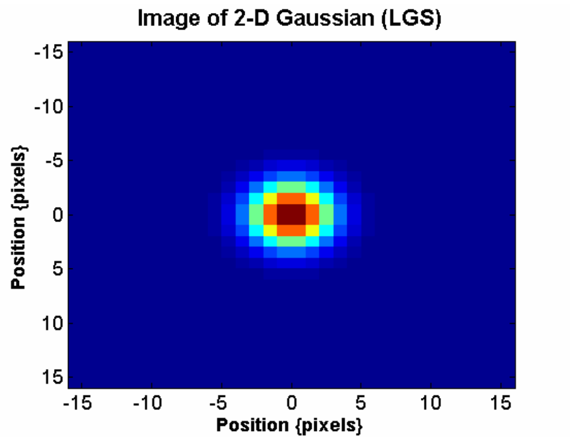


Figure 10. 2-D Gaussian and Vector Projections in x and y Planes

Figure 11. Observed Image of 2-D Gaussian without Noise

Aside from representing a smooth function from which over-sampling and interpolation are simple, a further benefit of using this model is the ease of creating projections of the image discussed later in Sub-Section 3.2.3. The Gaussian has an added advantage in that it is also a close representation of a diffraction limited natural guide star once the image shape adjusts to reflect the effects of noise, in which case a standard deviation of two accurately represents both the NGS and LGS for modeling purposes [4].

3.2.2. Using Real Images or Real Data

Use of real images requires more constraints than merely careful handling of imprecision in the Fourier transform, these types of images require proper centering to provide fair statistics, band-limiting to avoid aliasing, and down-sampling / up-sampling to meet sampling requirements. Centering typically requires use of an existing algorithm, such as the centroid, to adapt the image with either Fourier, or sinc, interpolation or another robust method such as bi-cubic or cubic-spline interpolation. Although sinc interpolation is ideal, this simulation uses MATLAB's cubic-spline interpolation for this step in modeling to prevent additional unnecessary information appearing in the black background needed for down-sampling and band-limiting operations of the image.

An important parameter that drives modeling is the pixel size relative to the actual size of the simulated object as defined by Equations 2 and 3. This modeling investigation seeks to use the Hubble satellite for large apertures on the order of one meter and subsequently a pixel size representing 20 cm [20]. Also of interest is a similar wave-front sensing problem in which the aperture reduces to 10 cm, forcing a corresponding change in pixel size to two meters. This pixel size determines the appropriate light-level for tracking or wave-front sensing for a real object as summarized in Equation 6 [4].

$$C = P_{sun} \cdot n \cdot A_{pixel} \cdot \frac{\Delta\lambda_{Sensor}}{\Delta\lambda_{Visible}} \cdot \frac{\pi\left(\frac{D}{2}\right)^2}{\pi \cdot z^2} \cdot \Delta t \cdot \frac{1}{h\nu} \cdot R \cdot B \qquad (6)$$

where

A_{pixel} = Area that a Pixel Represents on Object (m^2/pixel)

D = Diameter of the Aperture Opening (m)

n = Number of Pixels in Array (pixels)

P_{sun} = Power of Sun at Earth's Surface ≈ 1000 (W/m^2) [4]

$\Delta\lambda$ = Bandwidth of Light (Sensor ≈ $0.05\text{x}10^{-6}$, Visible ≈ $0.5\text{x}10^{-6}$) (m) [5]

z = Distance to the Object ≈ $600\text{x}10^3$ (m) [20]

Δt = Integration time of Imaging Device ≈ $100\text{x}10^{-6}$ (s) [5]

h = Planck's Constant ≈ $6.626\text{x}10^{-34}$ (J s) [18]

ν = Frequency of Light ≈ $6\text{x}10^{14}$ (Hz) [18]

R = Reflectance ≈ 5 to 20 (%) [18]

B = Light allocated by Beam-Splitter ≈ 10 (for WFS) (%) [5]

This equation involves several variables including the ratio of the aperture and distance to the object, integration time, photon energy, reflectance, and amount of light sent to the sensor, most of which are constant [4]. This formula sets the light-level values seen in Table 3 for the extended object scenario, and illustrates the light levels observed when viewing the Hubble space satellite. However, even with correct light-level calculations, a poor orientation of the image may result in low contrast for a particular dimension and a simple rotation of the image will alleviate a reduction in performance for all models.

To avoid aliasing while down-sampling, convolution in space with a sinc function, or multiplication by a two-dimensional rect function in frequency representing an ideal low-pass filter, removes high frequencies beyond the down-sampled image's bandwidth. If filtering did not occur, higher frequencies would alias to lower frequencies, corrupting the image in the spatial domain; this aliasing is a type of image corruption that up-sampling does not suffer. Down-sampling is straightforward for Nyquist sampled cases; however, other images, other samplings, or up-sampling requires sub-pixel information provided by an interpolator, and for speed in modeling this step uses MATLAB's bi-cubic interpolation after filtering.

The optical transfer function of a telescope further band-limits the image representing light passage through the particular optic in use and allowing for proper diffraction limiting effects caused by a fixed aperture. Using a standard diffraction limited OTF, with the factor to over-sample adjusting the aperture diameter directly, the impulse response, or magnitude-squared of the Fourier transform, is the point spread function, which is a two-dimensional Bessel function, or a perfect natural guide star [8]. Convolution of the image and PSF, or spatial frequency multiplication of the Fourier transform of the image and OTF, provides a properly band-limited image with the characteristics of the current aperture appropriately included. Throughout the down-sampling and band-limiting processes to create an image without noise for simulation, all real and complex data from the Fourier transforms passes through every step to limit errors due to imprecision in the Fourier transform. The images for wave-front sensing and tracking are visible in Figures 12 through 15.

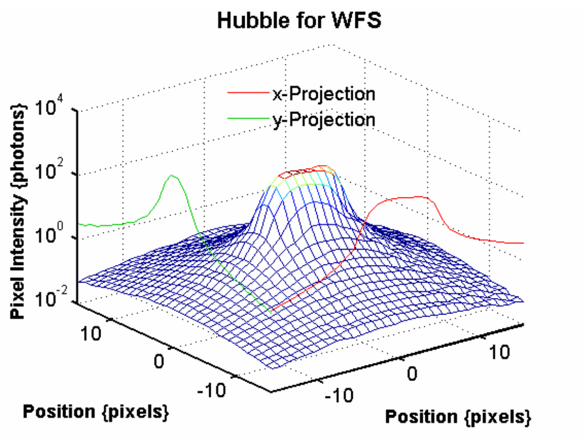


Figure 12. Hubble for Wave-Front Sensing & Projections in x and y Planes

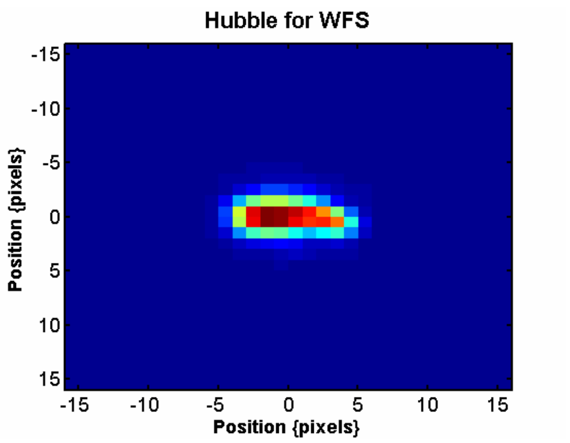


Figure 13. Observed Image of Hubble for Wave-Front Sensing without Noise

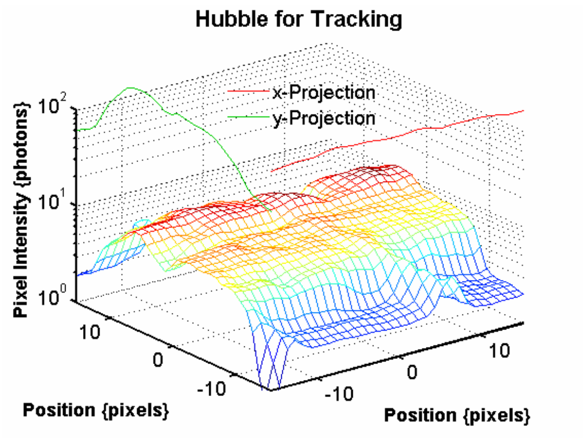


Figure 14. Hubble for Tracking and Projections in x and y Planes

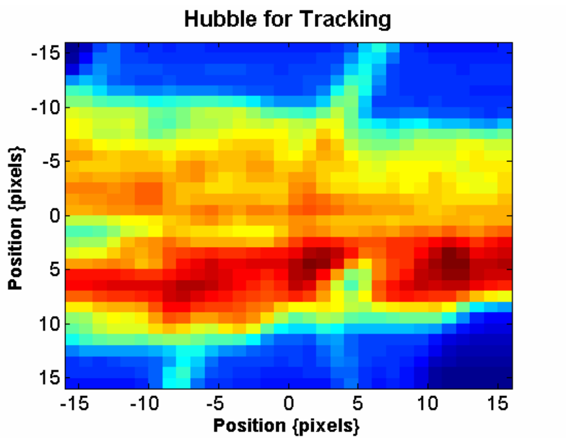


Figure 15. Observed Image of Hubble for Tracking without Noise

It is important to note that unless the ideal low-pass filter is the same or larger size of an up-sampled PSF, or zero-padded OTF, by the amount of down-sampling required, there are minor errors near the edge of the original down-sampled image, which is acceptable as long as the errors are outside of the window of interest used for simulation.

3.2.3. Image Projection / Vectorization

A projection of an image is purely the summation of an image in one dimension creating a vector representation of the image; and furthermore, this projection occurs after any cropping of the original image to maintain appropriate light-levels [5]. From a hardware viewpoint, this greatly increases the speed of image readout, which is the main limiting factor in the speed of closed-loop operation; unfortunately, this decreases the light-level by one-half as discussed in Sub-Section 3.5.2 [5].

Unrelated to noise statistics, an image projection implies independence between the two dimensions of an image, which is true for operations limited to projections of the entire image in a constant background as seen below in Theorem 1. To extend the applicability of this theorem, not only images wholly contained in a constant background but also images in a relatively low background with minor fluctuations exhibit dimensional independence; however, extended objects do not have dimensional independence as new information enters and exits the scene. Although this implies a requirement for joint estimation, the modeling here assumes dimensional independence as this is theoretically sufficient for simulation in a closed-loop environment [5].

Theorem 1

A projected image is independent with respect to the shift parameter in the orthogonal dimension as indicated by this formula:

$$i(x-\beta_x) = \int_{-\infty}^{\infty} i(x-\beta_x, y-\beta_y)dy$$

where

i	= Representation of 2-D Image (photons)
x, y	= Pixel Locations in the Image (pixels, $\in$ Integers)
β_x, β_y	= Shift in x or y direction (pixels, $\in$ Reals)

Proof

This first equation defines the starting point by demonstrating the two-dimensional inverse Fourier transform of an image with shifts in the x and y directions [11].

$$i(x-\beta_x, y-\beta_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(f_x, f_y) e^{-j2\pi(\beta_x f_x + \beta_y f_y)} e^{j2\pi(f_x x + f_y y)} df_x df_y$$

I	= 2-D Fourier Transform of Image
f_x, f_y	= Locations in Frequency Domain (frequency, $\in$ Integers)

Next is to integrate in the y dimension to produce a projection in the x dimension.

$$\int_{-\infty}^{\infty} i(x-\beta_x, y-\beta_y)dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(f_x, f_y) e^{-j2\pi(\beta_x f_x + \beta_y f_y)} e^{j2\pi(f_x x + f_y y)} df_x df_y dy$$

Separation of variables yields a smaller function integrated with respect to y.

$$\int_{-\infty}^{\infty} i(x-\beta_x, y-\beta_y)dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(f_x, f_y) e^{-j2\pi(\beta_x f_x + \beta_y f_y)} e^{j2\pi f_x x} \int_{-\infty}^{\infty} e^{j2\pi f_y y} dy df_x df_y$$

The integral on the right-hand-side equals a delta function.

$$\int_{-\infty}^{\infty} i\left(x-\beta_x, y-\beta_y\right)dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I\left(f_x, f_y\right)e^{-j2\pi\left(\beta_x f_x+\beta_y f_y\right)}e^{j2\pi f_x x}\delta\left(f_y\right)df_x df_y$$

Applying the sifting property of the delta function, this selects $f_y = 0$ in the integral.

$$\int_{-\infty}^{\infty} i\left(x-\beta_x, y-\beta_y\right)dy = \int_{-\infty}^{\infty} I\left(f_x, 0\right)e^{-j2\pi\beta_x f_x}e^{j2\pi f_x x}df_x$$

Since the right-hand-side is simply the inverse one-dimensional Fourier transform in the f_x direction with no dependence on the shift in the y direction, shifts in each dimension are indeed independent, which the two-dimensional Gaussian demonstrates further as its one-dimensional projection is simply a one-dimensional Gaussian.

Q.E.D.

3.3. Image Shifting

3.3.1. Shifting of a Known Function

The requirements for shifting a known function restrict modeling only to the point of requiring a shift of the function itself, which for a Gaussian is changing the mean of the function. By including the requirement of a continuous function, sub-pixel shifts, which are necessary to determine the true statistics of the model, are also straightforward.

3.3.2. FFT / Sinc-Interpolation

For images not generated from a smooth function, modeling requires another method for sub-pixel shifts, and the best method for sub-pixel shifting without knowledge of a function is through interpolation, and for small images sinc interpolation is the ideal method for accuracy. A good solution for implementing sinc interpolation is using the Fourier transform and the relationship between shifts in the spatial domain and phase shifts in the spatial frequency domain, as found in almost any discrete-time Fourier transform pair table, assuming circular shifts are acceptable or use of appropriate zero padding avoids circular shifting [11].

3.3.3. Sub-Pixel Shift Step Size

Although simulation step size typically determines smoothness of results, this modeling method requires different step sizes to illustrate different statistics representing the performance of the models appropriately. As a rule of thumb, the sub-pixel step size should be one-quarter to one-tenth the pixel size as a minimum for smooth results and no smaller than the interpolation search size used by the search algorithms and set by the CRLB. Any smaller simulation step size would provide no further insight beyond quantization error for bias calculations and no insight beyond noise error as indicated by the CRLB for noise calculations.

3.4. Calculating Bias and Mean Absolute Bias (MAB)

The error in the presence of no noise is difficult to remove and indicates the best possible operating characteristics of a sensor as well as the level of tuning required by the operator to achieve desirable statistics. To calculate bias, subtract the true shift value from the estimated shift value as indicated in Equation 7; however, bias can be deceiving, therefore it is better to compute absolute bias for comparison purposes.

$$AbsolueBias = \left|\hat{\beta}_{|NoNoise} - \beta\right| \tag{7}$$

where

$AbsoluteBias$ = The Absolute Value of the Error in No Noise (pixels, $\in$ Positive Reals)

$\beta_{|NoNoise}$ = The Estimated Shift Value without Noise (pixels, $\in$ Reals)

β = The True Shift Value (pixels, $\in$ Reals)

To reduce the complexity of results, averaging the x and y dimensions is acceptable as long as there is independence between these dimensions, otherwise unexpected results may surface. The bias should hold some similar properties to the image, in that if the image is symmetric the bias should be also, and if the image is higher contrast, the bias should indicate a relatively larger change in some areas of the curve if such a change is visible in the search window.

The average bias over a given number of pixels is the mean absolute bias (MAB) and provides a single number to describe operation of the sensor for a given region of the window. Regions of interest for the above images include an average over plus-or-minus four waves of tilt as this encompasses the entire window, and plus-or-minus one-half wave of tilt as this represents well over fifty percent of tilts seen in a closed loop system for reasonable values of D/r_0 as discussed previously in Sub-Section 3.1.3.

3.5. Noise Generation

3.5.1. Poisson and Bernoulli Random Variables

The statistics for light are intuitive from the packet perspective of light as each photon interacts with objects such as a beam splitter or charge couple device (CCD) as a Bernoulli random variable with a low rate of success. One way to approximate a Poisson random variable is to sum many Bernoulli trials, each with a low rate of success, hence, the overall statistics of light being approximately Poisson in nature [9]. Equation 8 is the general form of a Poisson random variable, and is the basis of the statistics required for modeling of noise for tracking and wave-front sensing. The expectation and variance statistics for Poisson random variables indicate how the SNR increases as the light intensity increases. Since the variance increases at the same rate as the mean, the standard deviation increases as the square-root of the mean, increasing the SNR in an initially logarithmic, and then nearly linear fashion as indicated by Figure 16. Over-sampling decreases the SNR on a per-pixel basis as the light splits between more pixels decreasing the available light per pixel. Since light-level also dictates the quality of a captured image, the lowest SNR possible for a wave-front sensor to operate properly is the ideal operating light-level and what this modeling attempts to parameterize.

$$p\left(d(x,y) \mid \left(i\left(x-\beta_x, y-\beta_y\right) \mid \beta_x, \beta_y\right)\right) = \frac{i\left(x-\beta_x, y-\beta_y\right)^{d(x,y)}}{d(x,y)!} e^{-i\left(x-\beta_x, y-\beta_y\right)} \tag{8}$$

where

d = The Observed Intensity (photons, $\in$ Integers)

i = The True Image Intensities (photons, $\in$ Reals)

x, y = Pixel Locations in the Image (pixels, $\in$ Integers)

β_x, β_y = Shift in x or y direction (pixels, $\in$ Reals)

$E[d]$ = i The True Image Intensity is the Mean (photons, $\in$ Reals)

$VAR[d]$ = i The True Image Intensity is the Variance (photons^2, $\in$ Positive Reals)

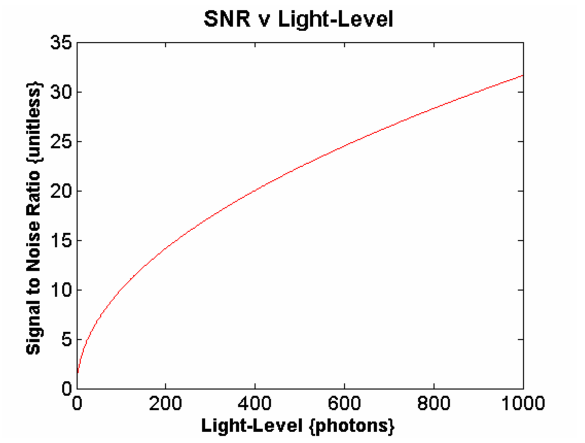


Figure 16. SNR v Light-Level

3.5.2. Effects of Projecting Images

With the exception of the Shack-Hartmann sensor model, all sensor models use vector, or projection, images, which cause some interesting effects for the noise statistics to compare properly between different sensors. Building on the assumption that each pixel is independent, one can show that the sum of Poisson random variables is a new Poisson random variable, with the new mean and variance being the sum of all means, using either the probability generating function, as shown in Theorem 2, or indirect convolution and knowledge of Taylor series expansions for an exponential.

Theorem 2

The summation of Poisson random variables, or convolution of probability mass functions, is another Poisson random variable with the new rate being the sum of the rates of the summed random variables:

$$\sum_{y} d(x,y) \to p(d(x) \mid (i(x-\beta_x) \mid \beta_x)) = p\left(\sum_{y} d(x,y) \mid \left(\sum_{y} i(x-\beta_x, y-\beta_y) \mid \beta_x, \beta_y\right)\right)$$

Proof

This first equation is merely the probability generating function redefined for the Poisson random variable used in this modeling and simulation [9]; note, Equation 8 defines all parameters except for z which is the transform variable.

$$G_{d(x,y)}(z) = e^{i(x-\beta_x, y-\beta_y)(z-1)}$$

Since a summation of random variables is really a convolution, this becomes a product in the z-domain as indicated below.

$$\prod_y G_{d(x,y)}(z) = \prod_y e^{i(x-\beta_x, y-\beta_y)(z-1)}$$

Now using exponential properties, the product becomes a summation in the exponent, and factoring out the (z-1) term puts the solution back into the original form.

$$\prod_y G_{d(x,y)}(z) = e^{\left(\sum_y i(x-\beta_x, y-\beta_y)\right)(z-1)}$$

This summation indicates the new rate, mean, and variance are simply the sum of intensities from the original image, providing an easy method of computing statistics for projected image data.

Q.E.D.

In addition to keeping the Poisson statistical nature for a projected image the light level decreases by half, which necessarily decreases the SNR as defined in the previous sub-section. By modeling two separate images at half intensity, including noise, then producing projection images for two-dimensions as well as summing both images together to form one, all sensors receive the same noise statistics while some operate on image projections and others operate on a complete image. This allows for accurate computation and comparison of statistics for simulation and development purposes.

3.5.3. Computing Noise Statistics

There are three main statistics for comparison between sensors when computing with noisy data; however, they are not unique and only two of them are useful to the developer and end user. The first two statistics are nearly the same, as one is simply the square of the other before averaging: mean absolute error (MAE) and mean square error

(MSE). To avoid confusion, the MSE used for modeling is the average square error and not the estimation technique familiar to researchers using signal processing estimation methods. To compute these statistics, refer to Equations 9 and 10, understanding that in a similar manner to MAB these statistics are clearer when averaged over a range of shift values such as one-half wave of tilt or four waves of tilt.

$$MAE = \frac{\sum_{Trials} \left| \hat{\beta}_{|Noise} - \beta \right|}{N} \tag{9}$$

where

MAE = Mean Absolute Error (pixels, $\in$ Reals)

$\hat{\beta}_{|Noise}$ = The Estimated Shift Value in Noise (pixels, $\in$ Reals)

β = The True Shift Value (pixels, $\in$ Reals)

N = The Number of Trials (unitless, Preferred to be a Power of 2)

$$MSE = \frac{\sum_{Trials} \left(\hat{\beta}_{|Noise} - \beta \right)^2}{N} \tag{10}$$

where

MSE = Mean Square Error (pixels^2, $\in$ Positive Reals)

Of the two statistics, MSE captures a broader view as it is a middle ground or combination of the MAB and VAR, as indicated by Equation 11, and is useful to see which of the two statistics drives the resulting performance of the sensor [19].

$$MSE \approx VAR + \left(Bias \right)^2 \tag{11}$$

where

VAR = Variance (pixels^2, $\in$ Positive Reals)

$Bias$ = Absolute Bias as Defined in Equation 7 (pixels, $\in$ Positive Reals)

Variance (VAR) as established in Equation 12 appears nearly the same as MSE with two significant differences: 1) the sample mean is the subtrahend rather than the true shift value, and 2) the divisor after summing the sample is one less than the total number of trials making it an unbiased estimate of the variance. As a second order statistic, VAR indicates how well a sensor can perform for a given noisy environment, and is impossible to remove without changing the type of estimation or optical setup.

$$VAR = \frac{\sum_{Trials}\left(\hat{\beta}_{|Noise} - \frac{\sum_{Trials}\hat{\beta}_{|Noise}}{N}\right)^2}{N-1} \tag{12}$$

Although only qualitative bounds are available for average bias and error, it is possible to provide an analytical bound for variance that defines the efficiency of an algorithm's ability to reject noise in various conditions. With the proper background and modeling capabilities, this Cramer-Rao lower bound can provide insight into development of an algorithm to improve tracking and wave-front sensing, while verifying simulation and experimental results.

3.6. Summary

Accurate modeling not only guides research to feasible solutions but also provides a method to verify research results before actual implementation. For this research effort, generation of images and the noise statistics that surround them is the key to better understanding and estimation of wave-front parameters and tracking shifts.

IV. Analysis

Investigation in wave-front sensing requires thorough knowledge of the current estimation techniques, environmental parameters, and modeling practices to provide guidance, insight, and validation capabilities for research. The key areas for investigation for this research include bounds on variance to quantify performance for any wave-front sensor, search algorithm optimization for the maximum-likelihood wave-front sensor to meet or exceed timing requirements, and an implementation proposal with hardware realization of the sensor algorithm to demonstrate feasibility of this implementation. Theory can provide excellent guidance for algorithm development and hardware implementation if applied correctly, as this research attempts to do; and the proper use of theoretical results can significantly shorten development time compared to trail and error analysis.

4.1. Cramer-Rao Lower Bound (CRLB) for Tilt Estimates Obtained with LGS

4.1.1. Relevant Statistics, Assumptions, and Setup

As the CRLB is a bound on variance, it requires statistical background and noise information given a particular type of data and a proper foundation to provide meaningful information. The basis of analysis resides with Equation 8 in Chapter III and the assumption of a form of the laser guide star for the image as a two-dimensional Gaussian represented by Equation 5; repeating both equations below provides clarity.

$$p\left(d(x,y) \mid \left(i\left(x-\beta_x, y-\beta_y\right) \mid \beta_x, \beta_y\right)\right) = \frac{i\left(x-\beta_x, y-\beta_y\right)^{d(x,y)}}{d(x,y)!} e^{-i\left(x-\beta_x, y-\beta_y\right)} \quad (8)$$

where

d = The Observed Intensity (photons, $\in$ Integers)

i = The True Image Intensities (photons, $\in$ Reals)

x, y = Pixel Locations in the Image (pixels, $\in$ Integers)

β_x, β_y = Shift in x or y direction (pixels, $\in$ Reals)

$$i(x,y) = C\left(2\pi\sigma^2\right)^{-1} e^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}} \quad (5)$$

where

i = Representation of 2-D Gaussian Image (photons)

x, y = Pixel Locations in the Image (pixels, $\in$ Integers)

C = Total Intensity of Image (photons)

σ = Standard Deviation (pixels, $\in$ Positive Reals)

Assuming $\sigma = \sigma_x = \sigma_y$ and the Two Dimensions are Independent

Assuming that the dimensions are independent, using the fact that the sum of Poisson random variables is another Poisson random variable, and using the image projection technique to remove one of the dimensions, Equations 8 and 5 become marginal with respect to x in Equations 13 and 14 below.

$$p\left(d(x) \mid \left(i\left(x-\beta_x\right) \mid \beta_x\right)\right) = \frac{i\left(x-\beta_x\right)^{d(x)}}{d(x)!} e^{-i\left(x-\beta_x\right)} \quad (13)$$

$$i(x) = C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-x^2}{2\sigma_i^2}} \quad (14)$$

Assuming the pixels for a projected image are independent this results in the joint probability mass function (PMF) representing the joint *a priori* density function in Equation 15.

$$\prod_x p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x\right)\right) = \prod_x \frac{i(x-\beta_x)^{d(x)}}{d(x)!} e^{-i(x-\beta_x)} \tag{15}$$

The random variables in this equation are the shift represented by β_x and two unwanted parameters C and σ_i, which are part of the assumed image form. However, this equation does not account for windowing of either the data or initial image as the information captured is finite in size, and the proposed maximum-likelihood sensor requires further windowing to search over shifts and to prevent detrimental effects from new data entering the scene [5]. To limit the product properly, a windowing function on both the true image as well as the captured image simply bounds the limits for the product function, and completes the probability information required to derive the CRLB for an unbiased estimator.

4.1.2. Derivation

As noted previously, estimation of the shift parameter is the goal; however, two additional parameters require estimation as well and therefore a joint estimation approach of these parameters and the CRLB serves as an accurate lower bound for Gaussian images. To derive a CRLB requires computation of the elements that compose the Fisher Information Matrix as defined in Equation 16, where the diagonal elements of the inverse of this matrix are the CRLB for the respective parameters in Equation 17 [19].

$$J_{ij} = -E\left[\frac{\partial^2 \ln p_{\vec{d}|\vec{a}}\left(\vec{D} \mid \vec{A}\right)}{\partial A_i \partial A_j}\right] \tag{16}$$

where

J = Elements of the Fisher Information Matrix

$\vec{D}$ = the Entire Vector *d(x)*

$\vec{A}$ = the Parameters to Estimate (β_x, σ_i, C)

$$\hat{\sigma}^2_{\vec{A}} \geq \boldsymbol{J}^{-1} \tag{17}$$

where

$\boldsymbol{J}$ = The Fisher Information Matrix

As this equation calls for the log of the joint likelihood, Equation 18 illustrates the log of Equation 15, with further simplifications.

$$\begin{aligned}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x\right)\right)\right) &= \sum_x \ln\left(\frac{i(x-\beta_x)^{d(x)}}{d(x)!} e^{-i(x-\beta_x)}\right) \\ &= \sum_x \left(\ln\left(i(x-\beta_x)^{d(x)}\right) - \ln\left(d(x)!\right) + \ln\left(e^{-i(x-\beta_x)}\right)\right) \\ &= \sum_x \left(d(x)\ln\left(i(x-\beta_x)\right) - \ln\left(d(x)!\right) - i(x-\beta_x)\right)\end{aligned} \tag{18}$$

To reflect the additional unwanted parameters, Equation 19 includes σ_i and C as additional givens in the log-likelihood, where the true image conditioned on these parameters represent the vector $\vec{A}$ in the Fisher Information Matrix and the observed data represents $\vec{D}$.

$$\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right) = \sum_x \left(d(x)\ln\left(i(x-\beta_x)\right) - \ln\left(d(x)!\right) - i(x-\beta_x)\right) \tag{19}$$

An assumption that windowing equation 19 does not change the derivative allows computation of partials without knowing the derivative of the window function; however, this assumption is only a close approximation when the observed image's intensity decreases to zero at the edge of the window making the CRLB applicable for images with a large black background or zero-shift estimation. This is the best-case operation of a sensor, and still provides an accurate lower bound for performance of estimation techniques. As the partial derivatives, logarithm, and partial derivatives of the log of the Gaussian image form appear several times in the next derivation, Equations 20 through 26 summarize these results based on Equation 14.

$$\frac{\partial}{\partial \beta_x} i(x-\beta_x) = i(x-\beta_x)\frac{(x-\beta_x)}{\sigma_i^2} \tag{20}$$

$$\frac{\partial}{\partial \sigma_i} i(x-\beta_x) = i(x-\beta_x)\left(\frac{(x-\beta_x)^2}{\sigma_i^3} - \frac{1}{\sigma_i}\right) = i(x-\beta_x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \tag{21}$$

$$\frac{\partial}{\partial C} i(x-\beta_x) = i(x-\beta_x)\frac{1}{C} \tag{22}$$

$$\ln\big(i(x-\beta_x)\big) = \ln\left(C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}\right) - \frac{(x-\beta_x)^2}{2\sigma_i^2} \tag{23}$$

$$\frac{\partial}{\partial \beta_x}\ln\big(i(x-\beta_x)\big) = \frac{(x-\beta_x)}{\sigma_i^2} \tag{24}$$

$$\frac{\partial}{\partial \sigma_i}\ln\big(i(x-\beta_x)\big) = \frac{(x-\beta_x)^2}{\sigma_i^3} - \frac{1}{\sigma_i} = \frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \tag{25}$$

$$\frac{\partial}{\partial C}\ln\big(i(x-\beta_x)\big) = \frac{1}{C} \tag{26}$$

Leveraging the information in Equations 20 through 26, the following equations compute the first partials of the log-likelihood, which are also useful for maximum-likelihood estimation of these parameters with the Gaussian image assumption.

$$\frac{\partial}{\partial \beta_x}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)$$

$$= \frac{\partial}{\partial \beta_x}\sum_x \left(d(x)\ln\left(i(x-\beta_x)\right) - \ln\left(d(x)!\right) - i(x-\beta_x)\right)$$

$$= \sum_x \left(\frac{\partial}{\partial \beta_x}\left(d(x)\ln\left(i(x-\beta_x)\right)\right) - \frac{\partial}{\partial \beta_x}\ln\left(d(x)!\right) - \frac{\partial}{\partial \beta_x} i(x-\beta_x)\right)$$

$$= \sum_x \left(d(x)\frac{(x-\beta_x)}{\sigma_i^2} - i(x-\beta_x)\frac{(x-\beta_x)}{\sigma_i^2}\right) \qquad (27)$$

$$\frac{\partial}{\partial \sigma_i}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)$$

$$= \frac{\partial}{\partial \sigma_i}\sum_x \left(d(x)\ln\left(i(x-\beta_x)\right) - \ln\left(d(x)!\right) - i(x-\beta_x)\right)$$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i} d(x)\ln\left(i(x-\beta_x)\right) - \frac{\partial}{\partial \sigma_i}\ln\left(d(x)!\right) - \frac{\partial}{\partial \sigma_i} i(x-\beta_x)\right)$$

$$= \sum_x \left(d(x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} - i(x-\beta_x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3}\right) \qquad (28)$$

$$\frac{\partial}{\partial C}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)$$

$$= \frac{\partial}{\partial C}\sum_x \left(d(x)\ln\left(i(x-\beta_x)\right) - \ln\left(d(x)!\right) - i(x-\beta_x)\right)$$

$$= \sum_x \left(\frac{\partial}{\partial C} d(x)\ln\left(i(x-\beta_x)\right) - \frac{\partial}{\partial C}\ln\left(d(x)!\right) - \frac{\partial}{\partial C} i(x-\beta_x)\right)$$

$$= \sum_x \left(d(x)\frac{1}{C} - i(x-\beta_x)\frac{1}{C} \right) \tag{29}$$

Unfortunately, the Fisher Information Matrix requires the second partials with respect to all estimated parameters; therefore, it is a square matrix and the elements of the matrix should be symmetric about the diagonal as the order of the partial derivatives should be reversible.

$$\frac{\partial^2}{\partial \beta_x^2} \sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)$$

$$= \frac{\partial}{\partial \beta_x} \sum_x \left(d(x)\frac{(x-\beta_x)}{\sigma_i^2} - i(x-\beta_x)\frac{(x-\beta_x)}{\sigma_i^2} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \beta_x}\left(d(x)\frac{(x-\beta_x)}{\sigma_i^2} \right) - \frac{\partial}{\partial \beta_x}\left(i(x-\beta_x)\frac{(x-\beta_x)}{\sigma_i^2} \right) \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \beta_x}\left(d(x)\frac{(x-\beta_x)}{\sigma_i^2} \right) - \left(\frac{(x-\beta_x)}{\sigma_i^2}\frac{\partial}{\partial \beta_x} i(x-\beta_x) + i(x-\beta_x)\frac{\partial}{\partial \beta_x}\frac{(x-\beta_x)}{\sigma_i^2} \right) \right)$$

$$= \sum_x \left(-d(x)\frac{1}{\sigma_i^2} - \left(i(x-\beta_x)\frac{(x-\beta_x)}{\sigma_i^2}\frac{(x-\beta_x)}{\sigma_i^2} - i(x-\beta_x)\frac{1}{\sigma_i^2} \right) \right)$$

$$= \sum_x \left(-d(x)\frac{1}{\sigma_i^2} - i(x-\beta_x)\left(\frac{(x-\beta_x)^2}{\sigma_i^4} - \frac{1}{\sigma_i^2} \right) \right) \tag{30}$$

$$\frac{\partial^2}{\partial \beta_x \partial \sigma_i} \sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)$$

$$= \frac{\partial}{\partial \sigma_i} \sum_x \left(d(x)\frac{(x-\beta_x)}{\sigma_i^2} - i(x-\beta_x)\frac{(x-\beta_x)}{\sigma_i^2} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i}\left(d(x)\frac{(x-\beta_x)}{\sigma_i^2} \right) - \frac{\partial}{\partial \sigma_i}\left(i(x-\beta_x)\frac{(x-\beta_x)}{\sigma_i^2} \right) \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i} \left(d(x) \frac{(x-\beta_x)}{\sigma_i^2} \right) - \left(\frac{(x-\beta_x)}{\sigma_i^2} \frac{\partial}{\partial \sigma_i} i(x-\beta_x) + i(x-\beta_x) \frac{\partial}{\partial \sigma_i} \frac{(x-\beta_x)}{\sigma_i^2} \right) \right)$$

$$= \sum_x \left(-d(x) \frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x) \left(\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \frac{(x-\beta_x)}{\sigma_i^2} - \frac{2(x-\beta_x)}{\sigma_i^3} \right) \right)$$

$$= \sum_x \left(-d(x) \frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x) \left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{(x-\beta_x)}{\sigma_i^3} - \frac{2(x-\beta_x)}{\sigma_i^3} \right) \right)$$

$$= \sum_x \left(-d(x) \frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x) \left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{3(x-\beta_x)}{\sigma_i^3} \right) \right) \qquad (31)$$

$$\frac{\partial^2}{\partial \beta_x \partial C} \sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C \right) \right) \right)$$

$$= \frac{\partial}{\partial C} \sum_x \left(d(x) \frac{(x-\beta_x)}{\sigma_i^2} - i(x-\beta_x) \frac{(x-\beta_x)}{\sigma_i^2} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial C} \left(d(x) \frac{(x-\beta_x)}{\sigma_i^2} \right) - \frac{\partial}{\partial C} \left(i(x-\beta_x) \frac{(x-\beta_x)}{\sigma_i^2} \right) \right)$$

$$= \sum_x \left(-i(x-\beta_x) \frac{1}{C} \frac{(x-\beta_x)}{\sigma_i^2} \right) \qquad (32)$$

$$\frac{\partial^2}{\partial \sigma_i \partial \beta_x} \sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C \right) \right) \right)$$

$$= \frac{\partial}{\partial \beta_x} \sum_x \left(d(x) \frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} - i(x-\beta_x) \frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \beta_x} \left(d(x) \frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) - \frac{\partial}{\partial \beta_x} \left(i(x-\beta_x) \frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) \right)$$

Substituting $U(\beta_x) = \dfrac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} = \dfrac{(x-\beta_x)^2}{\sigma_i^3} - \dfrac{1}{\sigma_i}$

$$= \sum_x \left(\frac{\partial}{\partial \beta_x}\left(d(x)U(\beta_x)\right) - \left(U(\beta_x)\frac{\partial}{\partial \beta_x} i(x-\beta_x) + i(x-\beta_x)\frac{\partial}{\partial \beta_x} U(\beta_x) \right) \right)$$

And Computing $\dfrac{\partial}{\partial \beta_x} U(\beta_x) = \dfrac{\partial}{\partial \beta_x}\dfrac{(x-\beta_x)^2}{\sigma_i^3} - \dfrac{\partial}{\partial \beta_x}\dfrac{1}{\sigma_i} = -\dfrac{2(x-\beta_x)}{\sigma_i^3}$

$$= \sum_x \left(-d(x)\frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x)\left(\left(\frac{(x-\beta_x)^2}{\sigma_i^3} - \frac{1}{\sigma_i} \right)\frac{(x-\beta_x)}{\sigma_i^2} - \frac{2(x-\beta_x)}{\sigma_i^3} \right) \right)$$

$$= \sum_x \left(-d(x)\frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x)\left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{(x-\beta_x)}{\sigma_i^3} - \frac{2(x-\beta_x)}{\sigma_i^3} \right) \right)$$

$$= \sum_x \left(-d(x)\frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x)\left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{3(x-\beta_x)}{\sigma_i^3} \right) \right) \tag{33}$$

$$\frac{\partial^2}{\partial \sigma_i^2}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)$$

$$= \frac{\partial}{\partial \sigma_i}\sum_x \left(d(x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} - i(x-\beta_x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i}\left(d(x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) - \frac{\partial}{\partial \sigma_i}\left(i(x-\beta_x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) \right)$$

Substituting $U(\sigma_i) = \dfrac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} = \dfrac{(x-\beta_x)^2}{\sigma_i^3} - \dfrac{1}{\sigma_i}$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i}\left(d(x)U(\sigma_i)\right) - \left(U(\sigma_i)\frac{\partial}{\partial \sigma_i} i(x-\beta_x) + i(x-\beta_x)\frac{\partial}{\partial \sigma_i} U(\sigma_i) \right) \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i}(d(x)U(\sigma_i)) - i(x-\beta_x)\left(U(\sigma_i)U(\sigma_i) + \frac{\partial}{\partial \sigma_i}U(\sigma_i) \right) \right)$$

And Computing $\frac{\partial}{\partial \sigma_i}U(\sigma_i) = \frac{\partial}{\partial \sigma_i}\frac{(x-\beta_x)^2}{\sigma_i^3} - \frac{\partial}{\partial \sigma_i}\frac{1}{\sigma_i} = \frac{1}{\sigma_i^2} - \frac{3(x-\beta_x)^2}{\sigma_i^4}$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i}(d(x)U(\sigma_i)) - i(x-\beta_x)\left(\left(\frac{(x-\beta_x)^2}{\sigma_i^3} - \frac{1}{\sigma_i} \right)^2 + \frac{1}{\sigma_i^2} - \frac{3(x-\beta_x)^2}{\sigma_i^4} \right) \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i}(d(x)U(\sigma_i)) - i(x-\beta_x)\left(\frac{(x-\beta_x)^4}{\sigma_i^6} - \frac{2(x-\beta_x)^2}{\sigma_i^4} + \frac{1}{\sigma_i^2} + \frac{1}{\sigma_i^2} - \frac{3(x-\beta_x)^2}{\sigma_i^4} \right) \right)$$

$$= \sum_x \left(d(x)\left(\frac{1}{\sigma_i^2} - \frac{3(x-\beta_x)^2}{\sigma_i^4} \right) - i(x-\beta_x)\left(\frac{(x-\beta_x)^4}{\sigma_i^6} - \frac{5(x-\beta_x)^2}{\sigma_i^4} + \frac{2}{\sigma_i^2} \right) \right) \quad (34)$$

$$\frac{\partial^2}{\partial \sigma_i \partial C}\sum_x \ln(p(d(x) \mid (i(x-\beta_x) \mid \beta_x, \sigma_i, C)))$$

$$= \frac{\partial}{\partial C}\sum_x \left(d(x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} - i(x-\beta_x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial C}\left(d(x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) - \frac{\partial}{\partial C}\left(i(x-\beta_x)\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) \right)$$

$$= \sum_x \left(-i(x-\beta_x)\frac{1}{C}\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) \quad (35)$$

$$\frac{\partial^2}{\partial C \partial \beta_x}\sum_x \ln(p(d(x) \mid (i(x-\beta_x) \mid \beta_x, \sigma_i, C)))$$

$$= \frac{\partial}{\partial \beta_x}\sum_x \left(d(x)\frac{1}{C} - i(x-\beta_x)\frac{1}{C} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \beta_x}\left(d(x)\frac{1}{C} \right) - \frac{\partial}{\partial \beta_x}\left(i(x-\beta_x)\frac{1}{C} \right) \right)$$

$$= \sum_x \left(-i(x-\beta_x)\frac{1}{C}\frac{(x-\beta_x)}{\sigma_i^2} \right) \tag{36}$$

$$\frac{\partial^2}{\partial C \partial \sigma_i} \sum_x \ln\left(p(d(x) \mid (i(x-\beta_x) \mid \beta_x, \sigma_i, C)) \right)$$

$$= \frac{\partial}{\partial \sigma_i} \sum_x \left(d(x)\frac{1}{C} - i(x-\beta_x)\frac{1}{C} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial \sigma_i}\left(d(x)\frac{1}{C} \right) - \frac{\partial}{\partial \sigma_i}\left(i(x-\beta_x)\frac{1}{C} \right) \right)$$

$$= \sum_x \left(-i(x-\beta_x)\frac{1}{C}\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3} \right) \tag{37}$$

$$\frac{\partial^2}{\partial C^2} \sum_x \ln\left(p(d(x) \mid (i(x-\beta_x) \mid \beta_x, \sigma_i, C)) \right)$$

$$= \frac{\partial}{\partial C} \sum_x \left(d(x)\frac{1}{C} - i(x-\beta_x)\frac{1}{C} \right)$$

$$= \sum_x \left(\frac{\partial}{\partial C}\left(d(x)\frac{1}{C} \right) - \frac{\partial}{\partial C}\left(i(x-\beta_x)\frac{1}{C} \right) \right)$$

$$= \sum_x \left(\frac{\partial}{\partial C}\left(d(x)\frac{1}{C} \right) - \left(\frac{1}{C}\frac{\partial}{\partial C} i(x-\beta_x) + i(x-\beta_x)\frac{\partial}{\partial C}\frac{1}{C} \right) \right)$$

$$= \sum_x \left(-d(x)\frac{1}{C^2} - \left(i(x-\beta_x)\frac{1}{C}\frac{1}{C} - i(x-\beta_x)\frac{1}{C^2} \right) \right)$$

$$= \sum_x \left(-d(x)\frac{1}{C^2} \right) \tag{38}$$

Because the second partial derivatives are interchangeable in order, this will create a symmetric Fisher Information Matrix as expected, further corroborating the results above.

The final step to complete the elements of the Fisher Information Matrix is to take the negative expectation of each second partial derivative, recognizing that the only random variable in Equation 19 is d(x), whose expectation is Equation 39.

$$E\left[d(x)\right] = i(x-\beta_x) = C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}} \tag{39}$$

$$\begin{aligned}
J_{11} &= -E\left[\frac{\partial^2 \ln p_{\vec{d}|\vec{a}}\left(\vec{D} \mid \vec{A}\right)}{\partial A_1{}^2}\right] \\
&= -E\left[\frac{\partial^2}{\partial \beta_x^2}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)\right] \\
&= -E\left[\sum_x\left(-d(x)\frac{1}{\sigma_i^2} - i(x-\beta_x)\left(\frac{(x-\beta_x)^2}{\sigma_i^4} - \frac{1}{\sigma_i^2}\right)\right)\right] \\
&= -\sum_x\left(-E\left[d(x)\frac{1}{\sigma_i^2}\right] - E\left[i(x-\beta_x)\left(\frac{(x-\beta_x)^2}{\sigma_i^4} - \frac{1}{\sigma_i^2}\right)\right]\right) \\
&= -\sum_x\left(-i(x-\beta_x)\frac{1}{\sigma_i^2} - i(x-\beta_x)\left(\frac{(x-\beta_x)^2}{\sigma_i^4} - \frac{1}{\sigma_i^2}\right)\right) \\
&= -\sum_x\left(-i(x-\beta_x)\frac{(x-\beta_x)^2}{\sigma_i^4}\right) \\
&= \frac{\sum_x\left(i(x-\beta_x)(x-\beta_x)^2\right)}{\sigma_i^4}
\end{aligned} \tag{40}$$

To put the result in Equation 40 in perspective, if the parameters σ_i and C are given, then the inverse of this would be the CRLB for the single parameter estimation; however, joint estimation requires the rest of the terms as well.

$$J_{12} = J_{21} = -E\left[\frac{\partial^2 \ln p_{\vec{d}|\vec{a}}\left(\vec{D} \mid \vec{A}\right)}{\partial A_1 \partial A_2}\right]$$

$$= -E\left[\frac{\partial^2}{\partial \beta_x \partial \sigma_i}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)\right]$$

$$= -E\left[\sum_x\left(-d(x)\frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x)\left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{3(x-\beta_x)}{\sigma_i^3}\right)\right)\right]$$

$$= -\sum_x\left(-E\left[d(x)\frac{2(x-\beta_x)}{\sigma_i^3}\right] - E\left[i(x-\beta_x)\left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{3(x-\beta_x)}{\sigma_i^3}\right)\right]\right)$$

$$= -\sum_x\left(-i(x-\beta_x)\frac{2(x-\beta_x)}{\sigma_i^3} - i(x-\beta_x)\left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{3(x-\beta_x)}{\sigma_i^3}\right)\right)$$

$$= \sum_x\left(i(x-\beta_x)\left(\frac{(x-\beta_x)^3}{\sigma_i^5} - \frac{(x-\beta_x)}{\sigma_i^3}\right)\right)$$

$$= \frac{\sum_x\left(i(x-\beta_x)(x-\beta_x)\left((x-\beta_x)^2 - \sigma_i^2\right)\right)}{\sigma_i^5} \tag{41}$$

$$J_{13} = J_{31} = -E\left[\frac{\partial^2 \ln p_{\vec{d}|\vec{a}}\left(\vec{D} \mid \vec{A}\right)}{\partial A_1 \partial A_3}\right]$$

$$= -E\left[\frac{\partial^2}{\partial \beta_x \partial C}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)\right]$$

$$= -E\left[\sum_x\left(-i(x-\beta_x)\frac{1}{C}\frac{(x-\beta_x)}{\sigma_i^2}\right)\right]$$

There are no random variables; therefore, the expectation has no effect.

$$= \frac{\sum_x \left(i(x-\beta_x)(x-\beta_x)\right)}{C\sigma_i^2} \tag{42}$$

$$J_{22} = -E\left[\frac{\partial^2 \ln p_{\vec{d}|\vec{a}}\left(\vec{D} \mid \vec{A}\right)}{\partial A_2{}^2}\right]$$

$$= -E\left[\frac{\partial^2}{\partial \sigma_i^2}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)\right]$$

$$= -E\left[\sum_x \left(d(x)\left(\frac{1}{\sigma_i^2} - \frac{3(x-\beta_x)^2}{\sigma_i^4}\right) - i(x-\beta_x)\left(\frac{(x-\beta_x)^4}{\sigma_i^6} - \frac{5(x-\beta_x)^2}{\sigma_i^4} + \frac{2}{\sigma_i^2}\right)\right)\right]$$

$$= -\sum_x \left(E\left[d(x)\left(\frac{1}{\sigma_i^2} - \frac{3(x-\beta_x)^2}{\sigma_i^4}\right)\right] - E\left[i(x-\beta_x)\left(\frac{(x-\beta_x)^4}{\sigma_i^6} - \frac{5(x-\beta_x)^2}{\sigma_i^4} + \frac{2}{\sigma_i^2}\right)\right]\right)$$

$$= -\sum_x \left(i(x-\beta_x)\left(\frac{1}{\sigma_i^2} - \frac{3(x-\beta_x)^2}{\sigma_i^4}\right) - i(x-\beta_x)\left(\frac{(x-\beta_x)^4}{\sigma_i^6} - \frac{5(x-\beta_x)^2}{\sigma_i^4} + \frac{2}{\sigma_i^2}\right)\right)$$

$$= \sum_x \left(i(x-\beta_x)\left(\frac{(x-\beta_x)^4}{\sigma_i^6} - \frac{2(x-\beta_x)^2}{\sigma_i^4} + \frac{1}{\sigma_i^2}\right)\right)$$

$$= \frac{\sum_x \left(i(x-\beta_x)\left((x-\beta_x)^2 - \sigma_i^2\right)^2\right)}{\sigma_i^6} \tag{43}$$

$$J_{23} = J_{32} = -E\left[\frac{\partial^2 \ln p_{\vec{d}|\vec{a}}\left(\vec{D} \mid \vec{A}\right)}{\partial A_2 \partial A_3}\right]$$

$$= -E\left[\frac{\partial^2}{\partial \sigma_i \partial C}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)\right]$$

$$= -E\left[\sum_x \left(-i(x-\beta_x)\frac{1}{C}\frac{(x-\beta_x)^2 - \sigma_i^2}{\sigma_i^3}\right)\right]$$

There are no random variables; therefore, the expectation has no effect.

$$= \frac{\sum_x \left(i(x-\beta_x)\left((x-\beta_x)^2 - \sigma_i^2\right)\right)}{C\sigma_i^3} \tag{44}$$

$$J_{33} = -E\left[\frac{\partial^2 \ln p_{\vec{d}|\vec{a}}\left(\vec{D} \mid \vec{A}\right)}{\partial A_3{}^2}\right]$$

$$= -E\left[\frac{\partial^2}{\partial C^2}\sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x, \sigma_i, C\right)\right)\right)\right]$$

$$= -E\left[\sum_x \left(-d(x)\frac{1}{C^2}\right)\right]$$

$$= -\sum_x \left(-E\left[d(x)\frac{1}{C^2}\right]\right)$$

$$= \sum_x \left(i(x-\beta_x)\frac{1}{C^2}\right)$$

$$= \frac{\sum_x i(x-\beta_x)}{C^2} \tag{45}$$

To summarize these results, Equation 46 displays the entire Fisher Information Matrix, and as Equation 17 illustrates, each diagonal element of the inverse of this matrix is the CRLB for the respective estimated parameters. The off-diagonal elements of the inverse Fisher Information Matrix represent the bounds on covariance terms determining independence, or lack of independence, between the estimated parameters.

$$\boldsymbol{J} = \begin{pmatrix} \frac{\sum_x i(x-\beta_x)(x-\beta_x)^2}{\sigma_i^4} & \frac{\sum_x i(x-\beta_x)(x-\beta_x)\left((x-\beta_x)^2-\sigma_i^2\right)}{\sigma_i^5} & \frac{\sum_x i(x-\beta_x)(x-\beta_x)}{C\sigma_i^2} \\ \frac{\sum_x i(x-\beta_x)(x-\beta_x)\left((x-\beta_x)^2-\sigma_i^2\right)}{\sigma_i^5} & \frac{\sum_x i(x-\beta_x)\left((x-\beta_x)^2-\sigma_i^2\right)^2}{\sigma_i^6} & \frac{\sum_x i(x-\beta_x)\left((x-\beta_x)^2-\sigma_i^2\right)}{C\sigma_i^3} \\ \frac{\sum_x i(x-\beta_x)(x-\beta_x)}{C\sigma_i^2} & \frac{\sum_x i(x-\beta_x)\left((x-\beta_x)^2-\sigma_i^2\right)}{C\sigma_i^3} & \frac{\sum_x i(x-\beta_x)}{C^2} \end{pmatrix} \tag{46}$$

This matrix is too complex to take the inverse of symbolically; however, assuming independence between parameters, as some off-diagonal elements' anti-symmetric nature indicates, may allow easy inversion of the diagonal elements. Figure 17 in the next sub-section illustrates the numerically calculated results for fixed parameters. Computing results numerically indicates two main points: for aliased images, small images, and near the edge of a fixed window, the bound appears incorrect; and a point solution for a zero-shift estimate appears valid for the majority of the window.

4.1.3. Simplification and Further Assumptions

It is possible to compute a zero-shift solution for the CRLB as the model is accurate for this condition since a zero-shift produces minimal discontinuities due to windowing in the derivative. Since the sampling of the Gaussian shape produces nearly linear regions between each sample, the summations over the values of x emulate integrals.

$$J_{11} \approx \frac{\int_{-\infty}^{\infty} \left(i(x-\beta_x)(x-\beta_x)^2 \right) dx}{\sigma_i^4}$$

$$= \frac{1}{\sigma_i^4} \int_{-\infty}^{\infty} \left(C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}} (x-\beta_x)^2 \right) dx$$

Select the following variables for integration by parts (IBP) and pull the constants out in front of the integral.

$U = (x-\beta_x)$, where $dU = dx$

$dV = (x-\beta_x)e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}dx$, where $V = -\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}$

$$= \frac{C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^4}\left(-(x-\beta_x)\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\Bigg|_{-\infty}^{\infty} - \int_{-\infty}^{\infty}\left(-\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx\right)$$

Attempting to evaluate the integral for U V yields ∞/∞; therefore, L'Hopital's Rule can still provide the limit as this function approaches ∞ in both directions.

$$\frac{(x-\beta_x)\sigma_i^2}{e^{\frac{(x-\beta_x)^2}{2\sigma_i^2}}}\Bigg|_{-\infty}^{\infty} = \frac{\infty}{\infty} - \frac{-\infty}{\infty}$$

$$\frac{\frac{d}{dx}(x-\beta_x)\sigma_i^2}{\frac{d}{dx}e^{\frac{(x-\beta_x)^2}{2\sigma_i^2}}} = \frac{\sigma_i^2}{\frac{(x-\beta_x)}{\sigma_i^2}e^{\frac{(x-\beta_x)^2}{2\sigma_i^2}}}$$

$$\frac{\sigma_i^2}{\frac{(x-\beta_x)}{\sigma_i^2}e^{\frac{(x-\beta_x)^2}{2\sigma_i^2}}}\Bigg|_{-\infty}^{\infty} = \frac{\sigma_i^2}{\infty\cdot\infty} - \frac{\sigma_i^2}{-\infty\cdot\infty} = 0$$

$$= \frac{C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^4}\int_{-\infty}^{\infty}\left(\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx$$

Rearranging this result produces a constant multiplied by the integral of a Gaussian, which is just the constant.

$$= \frac{C\sigma_i^2}{\sigma_i^4}\int_{-\infty}^{\infty}\left(\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx$$

$$= \frac{C}{\sigma_i^2} \tag{47}$$

$$J_{12} = J_{21} \approx \frac{\int_{-\infty}^{\infty}\left(i(x-\beta_x)(x-\beta_x)\left((x-\beta_x)^2-\sigma_i^2\right)\right)dx}{\sigma_i^5}$$

$$= \frac{1}{\sigma_i^5}\int_{-\infty}^{\infty}\left(i(x-\beta_x)\left((x-\beta_x)^3-\sigma_i^2(x-\beta_x)\right)\right)dx$$

$$= \frac{1}{\sigma_i^5}\left(\int_{-\infty}^{\infty}\left(i(x-\beta_x)(x-\beta_x)^3\right)dx - \sigma_i^2\int_{-\infty}^{\infty}\left(i(x-\beta_x)(x-\beta_x)\right)dx\right)$$

$$= \frac{1}{\sigma_i^5}\left(\int_{-\infty}^{\infty}\left((x-\beta_x)^3\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx - \sigma_i^2\int_{-\infty}^{\infty}\left((x-\beta_x)\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx\right)$$

$$= \frac{\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^5}\left(\int_{-\infty}^{\infty}\left((x-\beta_x)^2(x-\beta_x)e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx - \sigma_i^2\int_{-\infty}^{\infty}\left((x-\beta_x)e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx\right)$$

Selecting the following variables for IBP for the left-most integral creates a positive two times the integral on the right after one step of IBP, which combine to produce a single positive integral as shown below.

$U = (x-\beta_x)^2$, where $dU = 2(x-\beta_x)dx$

$dV = (x-\beta_x)e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}dx$, where $V = -\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}$

$$= \frac{\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^5}\left(\left(x-\beta_x\right)^2\left(x-\beta_x\right)e^{\frac{-\left(x-\beta_x\right)^2}{2\sigma_i^2}}\Bigg|_{-\infty}^{\infty} + \sigma_i^2\int_{-\infty}^{\infty}\left(e^{\frac{-\left(x-\beta_x\right)^2}{2\sigma_i^2}}\left(x-\beta_x\right)\right)dx\right)$$

$$= \frac{\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^5}\left(\frac{\left(x-\beta_x\right)^3}{e^{\frac{\left(x-\beta_x\right)^2}{2\sigma_i^2}}}\Bigg|_{-\infty}^{\infty} + \sigma_i^2\int_{-\infty}^{\infty}\left(\left(x-\beta_x\right)e^{\frac{-\left(x-\beta_x\right)^2}{2\sigma_i^2}}\right)dx\right)$$

Rather than laboriously apply L'Hopital's Rule three times to determine the limit of this fraction as it approaches ∞ in both directions, it is clear that the numerator will eventually be a constant and the denominator will remain an exponential dependent on x, again yielding 0 - 0.

$$\frac{\left(x-\beta_x\right)^3}{e^{\frac{\left(x-\beta_x\right)^2}{2\sigma_i^2}}}\Bigg|_{-\infty}^{\infty} = \frac{\infty}{\infty} - \frac{-\infty}{\infty} \Rightarrow \frac{6}{\infty} - \frac{6}{-\infty} = 0 - 0$$

$$= \frac{\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^5}\left(\sigma_i^2\int_{-\infty}^{\infty}\left(\left(x-\beta_x\right)e^{\frac{-\left(x-\beta_x\right)^2}{2\sigma_i^2}}\right)dx\right)$$

$$= \frac{\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^5}\left(\sigma_i^2\left(-\sigma_i^2 e^{\frac{-\left(x-\beta_x\right)^2}{2\sigma_i^2}}\right)\Bigg|_{-\infty}^{\infty}\right)$$

$$= \frac{\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{\sigma_i^5}\left(\sigma_i^2\left(-0+0\right)\right)$$

$$= 0 \qquad (48)$$

$$J_{13} = J_{31} \approx \frac{\int_{-\infty}^{\infty}\left(i\left(x-\beta_x\right)\left(x-\beta_x\right)\right)dx}{C\sigma_i^2}$$

$$= \frac{C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{C\sigma_i^2}\int_{-\infty}^{\infty}\left((x-\beta_x)e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx$$

$$= \frac{C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{C\sigma_i^2}\left(-\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\Bigg|_{-\infty}^{\infty}\right)$$

$$= \frac{C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}}{C\sigma_i^2}\left(-0+0\right)$$

$$= 0 \qquad (49)$$

$$J_{22} \approx \frac{\int_{-\infty}^{\infty}\left(i(x-\beta_x)\left((x-\beta_x)^2-\sigma_i^2\right)^2\right)dx}{\sigma_i^6}$$

$$= \frac{1}{\sigma_i^6}\int_{-\infty}^{\infty}\left(i(x-\beta_x)\left((x-\beta_x)^4-2\sigma_i^2(x-\beta_x)^2+\sigma_i^4\right)\right)dx$$

$$= \frac{1}{\sigma_i^6}\left(\int_{-\infty}^{\infty}\left((x-\beta_x)^4 i(x-\beta_x)\right)dx-2\sigma_i^2\int_{-\infty}^{\infty}\left((x-\beta_x)^2 i(x-\beta_x)\right)dx+\sigma_i^4\int_{-\infty}^{\infty}i(x-\beta_x)dx\right)$$

The right-hand integral is simply an integral of a constant multiplied by a Gaussian, which is the constant, whereas the center integral is identical to J_{11} and therefore equal to C σ_i^2. The left-hand integral requires temporarily pulling the constants out front and integration by parts as shown below.

$U=(x-\beta_x)^3$, where $dU = 3(x-\beta_x)^2\,dx$

$dV=(x-\beta_x)e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}dx$, where $V = -\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}$

$$\int_{-\infty}^{\infty}\left((x-\beta_x)^3(x-\beta_x)i(x-\beta_x)\right)dx$$

$$= C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}\left(-\left(x-\beta_x\right)^3\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\Bigg|_{-\infty}^{\infty} - \int_{-\infty}^{\infty}\left(-\sigma_i^2 e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}} 3\left(x-\beta_x\right)^2\right)dx\right)$$

Again, L'Hopital's Rule indicates the first term approaches zero; however, re-arranging the remaining integral reveals the same form as J_{11}, again simplifying the integration process by providing the answer of C σ_i^2.

$$\frac{\left(x-\beta_x\right)^3\sigma_i^2}{e^{\frac{(x-\beta_x)^2}{2\sigma_i^2}}}\Bigg|_{-\infty}^{\infty} = \frac{\infty}{\infty} - \frac{-\infty}{\infty} \Rightarrow \frac{6\sigma_i^2}{\infty} - \frac{6\sigma_i^2}{-\infty} = 0 - 0$$

$$= 3\sigma_i^2 C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}}\int_{-\infty}^{\infty}\left(\left(x-\beta_x\right)\left(x-\beta_x\right)e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx$$

$$= 3\sigma_i^2\left(C\sigma_i^2\right) = 3C\sigma_i^4$$

Inserting these results into the original equation yields the following:

$$= \frac{1}{\sigma_i^6}\left(3C\sigma_i^4 - 2C\sigma_i^4 + \sigma_i^4\right)$$

$$= \frac{2C}{\sigma_i^2} \tag{50}$$

$$J_{23} = J_{32} \approx \frac{\int_{-\infty}^{\infty}\left(i\left(x-\beta_x\right)\left(\left(x-\beta_x\right)^2 - \sigma_i^2\right)\right)dx}{C\sigma_i^3}$$

$$= \frac{1}{C\sigma_i^3}\int_{-\infty}^{\infty}\left(i\left(x-\beta_x\right)\left(x-\beta_x\right)^2 - i\left(x-\beta_x\right)\sigma_i^2\right)dx$$

$$= \frac{1}{C\sigma_i^3}\left(\int_{-\infty}^{\infty}\left(\left(x-\beta_x\right)\left(x-\beta_x\right)i\left(x-\beta_x\right)\right)dx - \sigma_i^2\int_{-\infty}^{\infty}\left(i\left(x-\beta_x\right)\right)dx\right)$$

The left-most integral is identical to the form found in J_{11}; therefore, using the same solution and L'Hopital's Rule calculation results in the following:

$$= \frac{1}{C\sigma_i^3}\left(\sigma_i^2 \int_{-\infty}^{\infty}\left(i(x-\beta_x)\right)dx - \sigma_i^2 \int_{-\infty}^{\infty}\left(i(x-\beta_x)\right)dx\right)$$

$$= \frac{1}{C\sigma_i^3}(0)$$

$$= 0 \tag{51}$$

$$J_{33} \approx \frac{\int_{-\infty}^{\infty} i(x-\beta_x)dx}{C^2}$$

$$= \frac{1}{C^2}\int_{-\infty}^{\infty}\left(C\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx$$

$$= \frac{C}{C^2}\int_{-\infty}^{\infty}\left(\left(2\pi\sigma_i^2\right)^{-\frac{1}{2}} e^{\frac{-(x-\beta_x)^2}{2\sigma_i^2}}\right)dx$$

$$= \frac{1}{C} \tag{52}$$

Equation 53 summarizes these results for the Fisher Information Matrix indicating that the magnitude of the true Fisher Information Matrix is less than or equal to this approximation to provide a true lower bound and that the parameters are uncorrelated, with the matrix inverse shown in Equation 54 providing an accurate approximation of the CRLB for all three parameters.

$$\boldsymbol{J} \lesssim \begin{pmatrix} \frac{C}{\sigma_i^2} & 0 & 0 \\ 0 & \frac{2C}{\sigma_i^2} & 0 \\ 0 & 0 & \frac{1}{C} \end{pmatrix} \tag{53}$$

$$CRLB: \begin{pmatrix} \sigma_{\hat{\beta}_x}^2 & 0 & 0 \\ 0 & \sigma_{\hat{\sigma}_i}^2 & 0 \\ 0 & 0 & \sigma_{\hat{C}}^2 \end{pmatrix} \gtrsim \begin{pmatrix} \frac{\sigma_i^2}{C} & 0 & 0 \\ 0 & \frac{\sigma_i^2}{2C} & 0 \\ 0 & 0 & C \end{pmatrix} \tag{54}$$

Figure 17 below indicates the match of these zero-shift solutions to the original numerically computed CRLB from Equation 46, which verifies the theoretical results, while Appendix A contains a Mathematica notebook to ensure every step is correct.

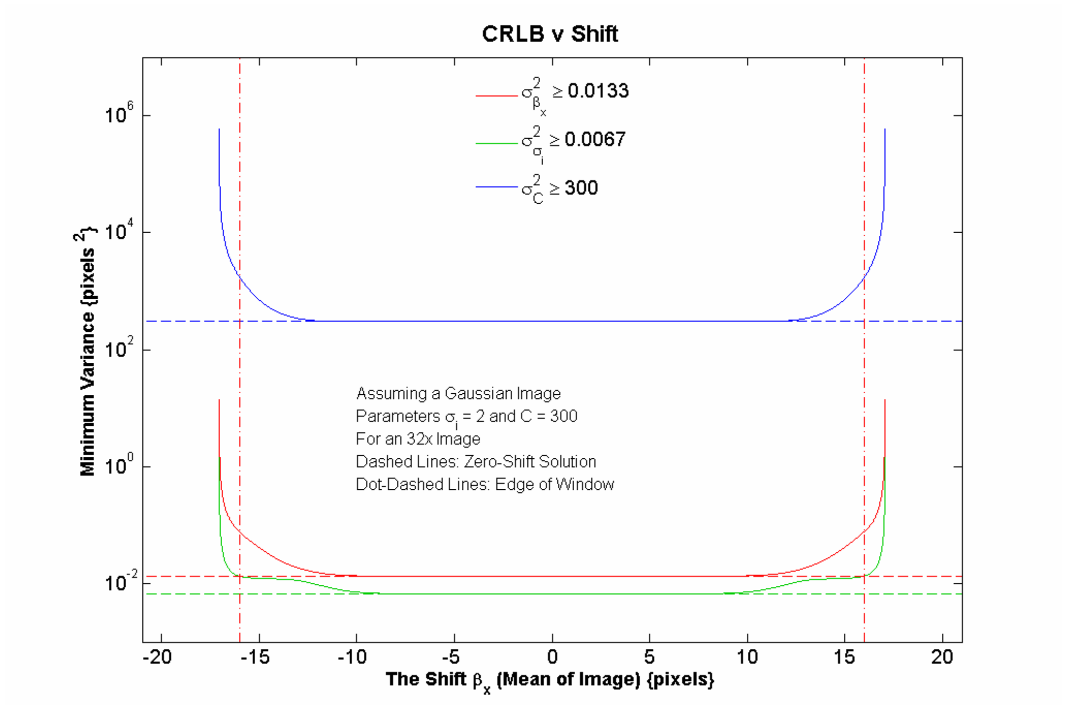


Figure 17. CRLB Numerical and Analytical Solution

Additionally, the last estimated parameter, C, directly stems from the variance of a Poisson random variable as summing all intensities in the vector produces this as the variance of the new Poisson random variable as described in the Section 3.5.

4.1.4. Benefits and Discussion

The above theoretical bound on variance for shift estimation has two main areas of benefit due to the simplicity and completeness of the bound; first, the bound can guide researchers in implementation of estimation algorithms, and second, it can guide technicians towards reasonable light-levels and images for shift estimation. The implementation benefits are two-fold in that estimation algorithms that search for sub-pixel shifts need only search to the square root of the minimum variance given by the bound as noise error overrides any quantization error in the model. The bound also provides an analytical method to validate the sensor model and simulation results by determining if the model is efficient in achieving the bound and providing another form of verification for modeling by allowing comparisons to this independent bound.

4.2. Maximum Likelihood Optimized Search Algorithm

4.2.1. Relevant Statistics, Assumptions, and Setup

Leveraging the noise statistics from Chapter III and the projection of an image exhibiting these statistics at the beginning of this chapter, the maximum-likelihood (ML) estimator uses the joint *a priori* distribution as shown in Equation 15 to determine what the estimate should be according to the criterion in Equation 55.

$$\hat{\beta}_{xML} = \arg\max \prod_{x} p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x\right)\right) \tag{55}$$

To minimize the computational complexity of performing numerous multiplications during a search over values of β_x, the same result is available from the natural log of Equation 55 as indicated in Equation 56.

$$\hat{\beta}_{xML} = \arg\max \sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x\right)\right)\right) \quad (56)$$

This maximum-likelihood approach differs from the maximum *a posteriori* (MAP) estimation approach, which seeks to maximize the likelihood over the joint *a posteriori* distribution found in Equation 57 by using either Bayes' Rule or applying the Law of Total Probability and the criterion found in Equation 58 [5].

$$\prod_x p\left(\left(i(x-\beta_x) \mid \beta_x\right) \mid d(x), \beta_x^{'}\right) = \prod_x \frac{p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x\right)\right) p\left(\left(i(x-\beta_x) \mid \beta_x\right) \mid \beta_x^{'}\right)}{p\left(d(x)\right)} \quad (57)$$

$$\hat{\beta}_{xMAP} = \arg\max \prod_x p\left(\left(i(x-\beta_x) \mid \beta_x\right) \mid d(x), \beta_x^{'}\right) \quad (58)$$

Since this technique seeks to maximize the joint *a posteriori* distribution for a given value of β_x, the marginal with respect to the observed image is unnecessary, while using the log of the *a posteriori* distribution and expanding further simplifies the search as indicated by Equation 59.

$$\hat{\beta}_{xMAP} = \arg\max \sum_x \ln\left(p\left(d(x) \mid \left(i(x-\beta_x) \mid \beta_x\right)\right)\right) + \sum_x \ln\left(p\left(\left(i(x-\beta_x) \mid \beta_x\right) \mid \beta_x^{'}\right)\right) \quad (59)$$

In the case that the right-hand term, or the prior probability of β_x given the previous shift, is uniform, the dependence on the prior withdraws causing the MAP and ML estimates to become equal. As mentioned in the chapter on modeling, this prior distribution is Gaussian in nature; however, the parameters required for this distribution are not available leaving the choice of assuming a uniform distribution, as previously performed in literature [4, 5].

The expansion of Equation 56 yields the log-likelihood for implementation and the first part of the MAP estimator, should the prior information become available, as specified by Equation 60.

$$\hat{\beta}_{x\,ML} = \arg\max \sum_x \ln\left(\frac{i(x-\beta_x)^{d(x)}}{d(x)!} e^{-i(x-\beta_x)} \right)$$

$$= \arg\max \sum_x \left(\ln\left(i(x-\beta_x)^{d(x)} \right) + \ln\left(e^{-i(x-\beta_x)} \right) - \ln(d(x)!) \right)$$

$$= \arg\max \sum_x \left(d(x)\ln(i(x-\beta_x)) - i(x-\beta_x) - \ln(d(x)!) \right)$$

Since the final term does not depend on β_x, the term drops out.

$$= \arg\max \sum_x \left(d(x)\ln(i(x-\beta_x)) - i(x-\beta_x) \right) \tag{60}$$

Ideally, the most efficient method for obtaining the shift estimate is through taking the derivative of the above function, setting it to zero, solving for the nodes of the function, and determining which node has the largest peak. It may be possible to derive a closed-form solution for the derivative of the log-likelihood provided the derivative of the original image $i(x\text{-}\beta_x)$ also has a closed form solution, which is both image and shift specific. This typically is not possible; however, leveraging the unique properties of the Fourier Transform and the interchangeability of derivatives and integrals with additional assumptions may provide such a closed form solution for faster estimation.

There are numerous alternative approaches to searching the log-likelihood including using a known function for the true image to allow use of all of the data in the log-likelihood calculation. Although a known function could preclude the need to window the data, simply adjusting the algorithm to accommodate both a fixed true image size and a dynamic true image size allows a greater search area for smaller image sizes and possibly better performance without modification to most software or hardware. An additional assumption when working with actual data is regarding which projected image to shift for searching over different search estimates, as it is possible to shift either the true image or the observed data. Theory indicates that the random parameter is in the original image, and therefore shifting of the true image is appropriate; however, it may be interesting to characterize the noise rejection capabilities of shifting the observed data also, as hardware interpolation is possible for this sub-pixel shift technique. This research focuses on implementing a maximum-likelihood search approach using the joint log-likelihood defined by Equation 60 in an efficient manner to temporally compete with the Shack-Hartmann and SWAT wave-front sensors, as current research indicates a statistical performance improvement with the ML sensor for extended objects [5].

4.2.2. Properties of Log-Likelihood Leveraged

As observed from computing sample log-likelihoods using the modeling techniques in Chapter III, there are several properties of the log-likelihood curve that lend themselves to an optimized search algorithm. Figure 18 and Figure 19 illustrate the log-likelihood curve for the laser guide star as described in the modeling chapter. The most important attributes include the large main node, significantly smaller nodes and distortion due to noise, and mild peaks at the end-points indicating performance in a constant background.

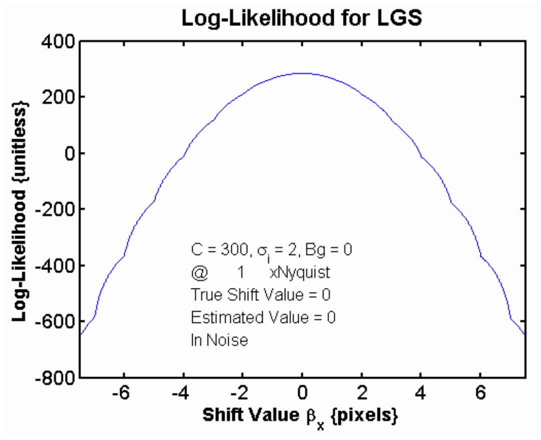


Figure 18. Log-Likelihood for Gaussian in Noise

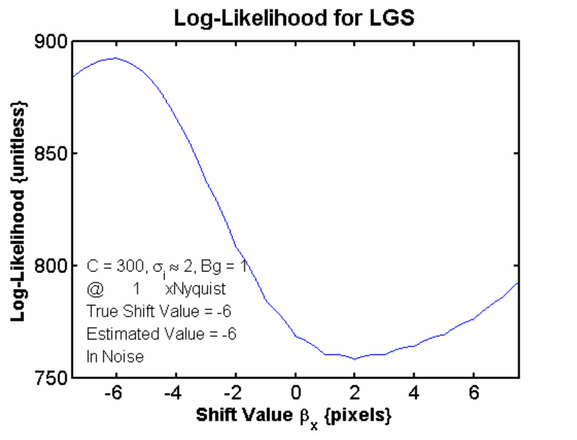


Figure 19. Log-Likelihood for Shifted Gaussian in Background and Noise

The peak of the main node is the estimate, and a search algorithm that rejects the noise and other characteristics of the log-likelihood could search in an efficient manner using the concave-down properties of the main node. The peaks at the edge of the log-likelihood window develop when the black background of the true image covers greater than one-half of the search window making it more likely that the object has moved completely out of view. These end-point peaks have the unique characteristic of being slightly greater on the side of the log-likelihood curve that contains the main node for a significant shift, also lending possible simplification to the search algorithm.

4.2.3. Search Algorithm Definition

The goal of this search algorithm is to perform an efficient search of the concave-down portion of the main node of the log-likelihood, while being robust enough to reject interference from noise and other artifacts in the search window. A method that skips the noise and artifacts by quickly finding the main node before performing finely stepped search operations effectively meets these requirements and provides a robust solution for the search algorithm. By pre-computing the interpolated true image and its logarithm, each sensor can use this data without wasting more computations allowing the optimized algorithm to focus on searching the log-likelihood and ignoring the requirements of the input data. The implemented search algorithm has two phases; the first grid search is optional depending on the shape of the log-likelihood and the second implements an optimal search algorithm requiring minimal memory storage for a known concave down function as indicated by the program flow in Figure 20.

There are two different ways to describe the main search algorithm using modern search techniques; the first method stems from the Gradient Decent algorithm, while the second method builds upon the algorithmic concept of a Binary Search Tree. From the Gradient Decent perspective, which is the basis for development, this algorithm performs Gradient Ascent by climbing the log-likelihood curve, where the step size and slope determinations are the unique and key components of the algorithm. The step size uses Bisection to determine the next point in the search, as it is easy to compute this dynamically changing step size and reduces the complexity of the search significantly. The slope determination ensures locating the peak by determining which direction to climb when encountering a larger log-likelihood value with version 2 of the Select Window Endpoints block choosing the new search region. The algorithm properly assumes that the slope is toward the current largest value if the new log-likelihood of the computed point is less than the current maximum and performs endpoint detection using version 3 of the Select Window Endpoints block.

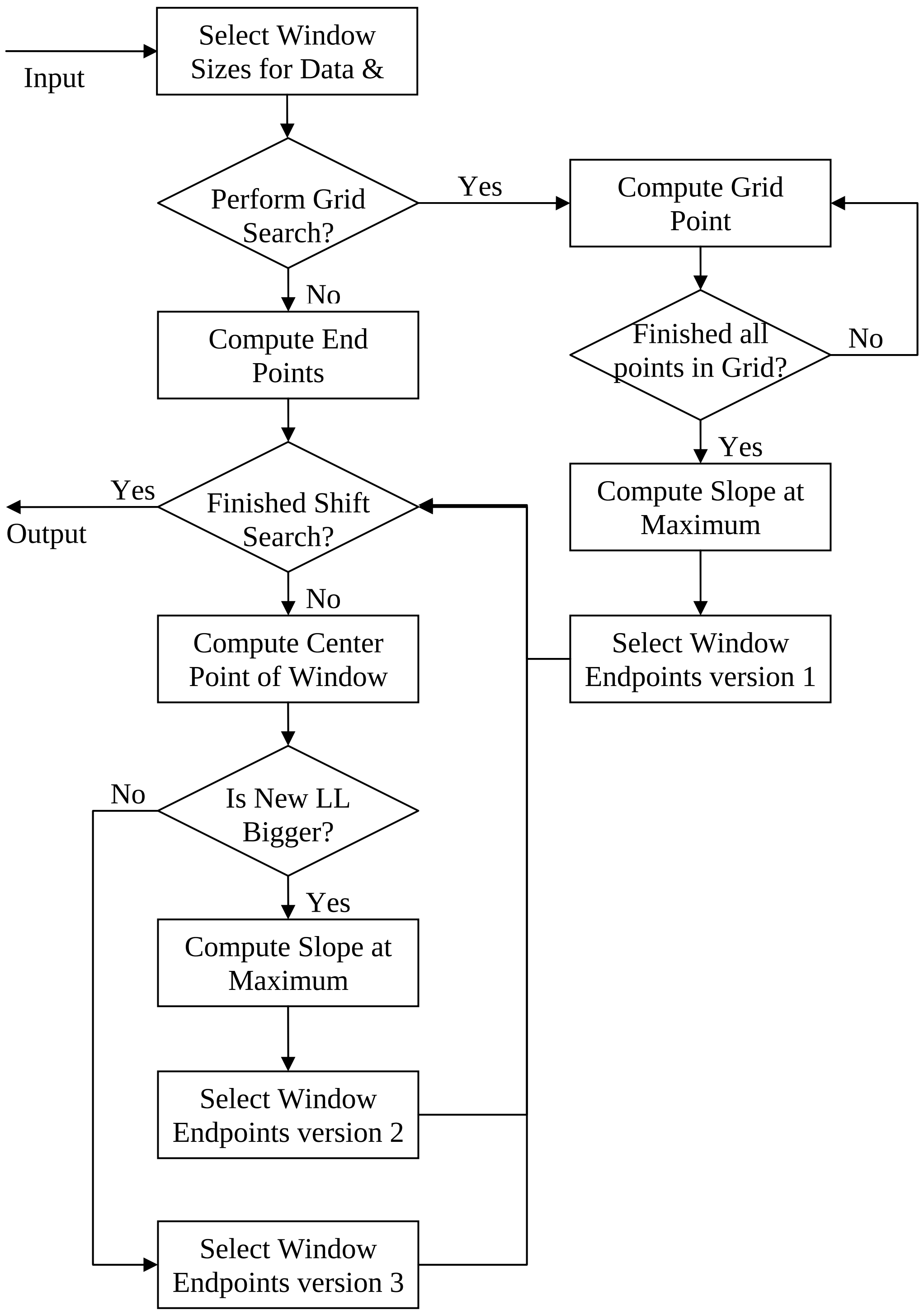

Input
Select Window Sizes for Data &
Perform Grid Search?
Yes
No
Compute Grid Point
Finished all points in Grid?
No
Yes
Compute End Points
Compute Slope at Maximum
Finished Shift Search?
Yes
Output
No
Select Window Endpoints version 1
Compute Center Point of Window
Is New LL Bigger?
No
Yes
Compute Slope at Maximum
Select Window Endpoints version 2
Select Window Endpoints version 3

Figure 20. Flow Diagram of Optimized Log-Likelihood Search Algorithm

This slope determination allows the search to narrow the search region for the peak quickly and completes the Gradient Decent algorithm. From the perspective of dividing the problem into search regions, another view of the algorithm emerges in the form of a Binary Search Tree, with the first node being the entire search window, this node's children being the left and right halves of the search window, and continuing until only individual elements are the leaves of the tree. As the search progresses the algorithm makes a decision at each node to guide which children to select and proceeds with a depth-first search of the entire tree; and since these decisions are final, upon reaching a leaf, the index of the leaf is the result of the search algorithm. This search algorithm has the added advantage that it requires memory for only three index values, log-likelihood values, and their attributes making it very feasible for implementation in a compact embedded architecture for fast operation.

The optional grid search allows the algorithm to proceed given the main node of the log-likelihood cannot be found within the first three computations of the bisection-based search algorithm. This search is simply a linear search across the window, with the results guaranteeing a narrowed search region decided by version 1 of the Select Window Endpoints block including only the main node of the log-likelihood curve. The nature of this search requires one additional memory location to ensure proper capture of the maximum log-likelihood and the window surrounding this maximum, but it remains simple due to capturing the maximum while computing the new log-likelihood values and performing only one slope detection when complete.

These algorithms require knowledge of the size of the observed image, or data, and true image arrays as well as the number of sub-pixel points interpolated between the points in the real data; however, they do not require knowledge of the type of interpolator used, which Appendix B further confirms. The number of sub-pixel points between real points is a direct result from the CRLB analysis, as selecting an interpolation allowing searches smaller than the minimum standard deviation simply increases computation time with no additional accuracy benefits. Using twice the light-levels suggested in the chapter on modeling as well as the assumption of a standard deviation of two for a Gaussian image results in a minimum standard deviation of error for shift estimation on the order of 0.03 pixels. Interpolating by 2^5 points narrowly accommodates this value, while 2^6 sufficiently surpasses this minimum error; therefore, a step size of the inverse of 2^7 represents an ideal interpolation level to allow for proper estimation of any type of image while only requiring one additional log-likelihood computation. Since each sensor requires only one true image as indicated in Section 4.3, a separate processor or module optimized for interpolation could provide this information. Research indicates that linear interpolation is good enough for estimation; however, a characterization of the performance difference between linear and the best possible cubic-spline interpolator that MATLAB includes is an interesting area of investigation included in this research [5].

4.2.4. Computational Complexity Analysis and Comparison

Computational complexity research provides insight into how difficult or how long an algorithm takes compared to another algorithm given the same inputs, and the following analysis performs this comparison for the Shack-Hartmann, SWAT, non-optimized Maximum-Likelihood, and optimized Maximum-Likelihood wave-front sensors. This is a comparison of the software portion of the algorithm only; the delays associated with hardware CCD readout and data transfer do not appear in these computations or comparisons and are important considerations outside of this discussion. The standard notation of $O_n(n)$ describes the computational complexity by indicating the asymptotic nature of a function based on the subscripted variable. To clarify the calculations, each sensor's computational complexity stems from a single-dimension estimate of the wave-front tilt, with significant constants indicated in the standard notation with a '*' for multiplication or '+' for addition between the two independent complexities. The Shack-Hartmann wave-front sensor requires software projection of the image when computing the centroid; therefore, this excess computation time appears in the number of additions required for an image as indicated by Table 4. The faster SWAT wave-front sensor capitalizes on hardware projection of the image, which also decreases the read-out time of the data, and shows a linear growth of both additions and multiplications with image size. The far more complex maximum likelihood algorithm breaks into three phases, with one optional phase, where the summation of these three phases indicates the total complexity of the search algorithm. Assuming a fixed window of one-half the image length, the complexity of the pre-compute phase and a non-optimized search algorithm appears in Table 5, while Table 6 presents the complexity of the optimized search algorithm for the optional grid search and gradient decent algorithm.

Table 4. Algorithm complexity for Shack-Hartmann and SWAT Sensors

Sensor	Shack-Hartmann		SWAT	
Type of Operation	Complexity n = image length	$O_n(\,)$	Complexity n = image length	$O_n(\,)$
Additions	$2n^2+n-2$	$O_n(n^2)$	$2n-2$	$O_n(n)$
Multiplications	n	$O_n(n)$	n	$O_n(n)$
Divisions	1	$O_n(1)$	1	$O_n(1)$

Table 5. Algorithm complexity for Maximum-Likelihood Sensor Phase I

Operation	Phase I		Non-Optimized Search	

	Complexity
	n = image length
	N = # to interpolate
		$O_n()(*,+)$
		$O_N()$
			Complexity
			n = image length
			N = # to interpolate
				$O_n()(*,+)$
				$O_N()$

Logical Shift	(n-1)(N-1)	$O_n(n)$* $O_N(N)$	1	$O_n(1)$ $O_N(0)$
Additions	(n-1)(N-1)	$O_n(n)$* $O_N(N)$	(n-1)((n/2)N+1)+(nN+2)	$O_n(n^2)$* $O_N(N)$
Multiplications	0	$O_n(0)$ $O_N(0)$	(n/2)((n/2)N+1)	$O_n(n^2)$* $O_N(N)$
Natural Logs	(n-1)N+1	$O_n(n)$* $O_N(N)$	0	$O_n(0)$ $O_N(0)$
Divisions	0	$O_n(0)$ $O_N(0)$	0	$O_n(0)$ $O_N(0)$
Log-Likelihood Computations	0	$O_n(0)$ $O_N(0)$	(n/2)N+1	$O_n(n)$* $O_N(N)$

Table 6. Algorithm complexity for Maximum-Likelihood Sensor Phases II and III

Where

g' = 1 if including the optional grid search, 0 otherwise

g = the number of points in the optional search grid, or one

Operation	Phase II		Phase III	

	Complexity
	n = image length
	g = # of Grid points
	N = # to interpolate
		$O_n()(*,+)$
		$O_N()$
			Complexity
			n = image length
			g = # of Grid points
			N = # to interpolate
				$O_n()(*,+)$
				$O_N()$

Logical Shift	0	$O_n(0)$ $O_N(0)$	1	$O_n(1)$ $O_N(0)$
Additions	3(n/2)g	$O_n(n)$ $O_N(0)$	Best: $(n+1)(2-2g'+\log_2((n/(2g))N))$ Worst: $(n+1)(2-2g'+2\log_2((n/(2g))N))$	$O_n(n\log(n))+$ $O_N(\log(N))$
Multiplications	(n/2)g	$O_n(n)$ $O_N(0)$	Best: $(n/2)(2-2g'+\log_2((n/(2g))N))$ Worst: $(n/2)(2-2g'+2\log_2((n/(2g))N))$	$O_n(n\log(n))+$ $O_N(\log(N))$
Natural Logs	0	$O_n(0)$ $O_N(0)$	0	$O_n(0)$ $O_N(0)$
Divisions	0	$O_n(0)$ $O_N(0)$	0	$O_n(0)$ $O_N(0)$
Log-Likelihood Computations	G	$O_n(1)$ $O_N(0)$	Best: $2-2g'+\log_2((n/(2g))N)$ Worst: $2-2g'+2\log_2((n/(2g))N)$	$O_n(\log(n))+$ $O_N(\log(N))$

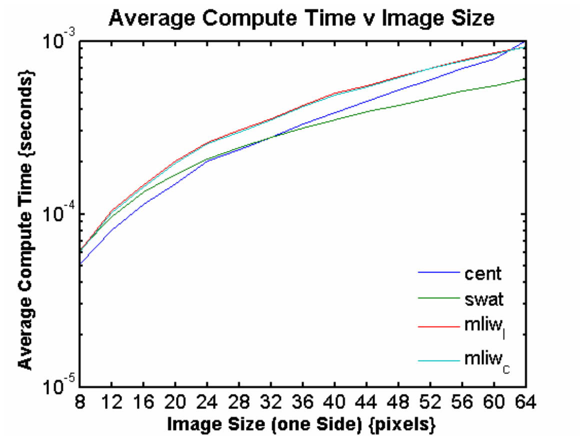


Figure 21. Illustration of Log-Linear Nature of ML Algorithm for LGS

To illustrate these results for varying image sizes, Figure 21 provides MATLAB simulation results, using floating-point computations on an Athlon 2600+ processor, of average time required for Shack-Hartmann sensor, cent on the plot, SWAT wave-front sensor, and the optimized maximum-likelihood algorithm (mliw with linear and MATLAB's cubic-spline interpolation) for different image sizes. The pre-compute phase requires the same computational complexity for that of SWAT estimation for additions but subsequently includes the additional time for computing logarithms as well. The non-optimized log-likelihood search algorithm clearly exceeds the computational complexity of both centroiding sensors; however, the optimized algorithm is only slightly more complex compared to the SWAT wave-front sensor. Provided the pre-compute time occurs off-line or over the span of a few estimates, the optimized algorithm performs well, with O(n log(n)) complexity, which is far superior to the non-optimized search algorithm.

4.2.5. Possible Improvements

Through investigation of the requirements and assumptions for the optimized search algorithm to perform properly, a few improvements are possible given thorough understanding of the algorithm and the data inputs required. This optimized algorithm assumes that the interpolated and projected true image, the log of this projection, and the observed data are available in memory for multiple accesses, with any pre-computation completed before estimation begins. Overall performance improvement may be possible by computing on-the-fly interpolation of the true image, which also requires on-the-fly logarithms, allowing for a slightly decreased external storage space, greatly decreased data access times, but more computations overall. If implemented, on-the-fly interpolation could lead to automatic interpolation level, or sub-pixel step size, selection that could further reduce the number of computations required for the search algorithm.

Although this optimized search algorithm is computationally efficient for the number of log-likelihood calculations performed, any other search or sort algorithm capable of exploiting new hardware technologies and possibly breaking apart the log-likelihood calculation itself may improve upon this search algorithm. For much larger image sizes, and possibly joint estimation in two dimensions, an artificial intelligence algorithm could provide faster results.

4.2.6. Limitations

The pre-computation of the interpolation and logarithm of the true image could significantly hinder the effectiveness of this wave-front sensing algorithm depending on the nature of the observed image, as this true image requires updates to prevent changes in light-level, orientation, and contrast from affecting the outcome of the log-likelihood.

Additionally, the implemented version of the algorithm requires the image size be a multiple of four, which reduces the possible array sizes for a given AO hardware setup.

4.3. Implementation Strategy

4.3.1. System Layout

The benefit of a maximum-likelihood vector-correlating wave-front sensor is the ability to perform tracking and wave-front sensing for AO using existing technologies with hardware requirements that typical systems already meet. As long as the imaging device meets or exceeds Nyquist sampling criteria, and the image provided contains twice or greater the number of pixels needed for the search window with appropriate light-level and contrast, nearly any imaging system can use this algorithm as a tracking or wave-front sensor. Since an appropriate hardware layout exists (lens, array, CCD, etc) for this modern wave-front sensor, this section focuses on the efficiency of the charge-coupled device (CCD) array, the setup of the processing unit hardware, and method to implement the maximum-likelihood estimation algorithm [5].

The size of the image follows from the shift detection requirements; however, exceeding this can greatly reduce the bias and overall error in estimates. Thus an image larger than thirty-two by thirty-two pixels is the recommended size for tracking and wave-front sensing of plus or minus four-waves of tilt, corresponding to a CCD array of 160 or greater pixels for an array of five-by-five wave-front sensors. An important possibility to consider if timing constraints permit is multiplexing the x and y dimension estimates in time, allowing for twice the light per dimension and requiring only once CCD array instead of two. If the CCD array has a fast analog to digital converter, it could interpolate the data directly for searching of shifts in the observed image instead of the true image, which offloads this requirement from the pre-compute phase.

The basic setup remains the same except for the above-mentioned improvements and the specification of a field programmable gate array (FPGA) programmed to execute the pre-compute phase and search algorithm. The targeted architecture for this application is an Altera Cyclone II FPGA, which is not the fastest or most complex FPGA, but it includes some basic modules to aid in computation, a reasonable package size, and a greatly reduced cost [7]. The development environment for this device is free and allows the researcher to validate implemented algorithms in the targeted environment with a simple and easy to use interface.

4.3.2. VHDL Implementation

The approach to algorithm implementation in this Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) follows a basic strategy to break the problem into smaller parts and implement them one at a time, which requires an extensible state-machine design with robust transitions and outputs. Abstraction is critical to reduce repeated computations to manageable modules, prevent excessively long code, and allow for easier debugging. The above algorithm reduces easily to the Moore state-machine in Figure 22, allowing for robust transitions, controlled outputs, and extensions to include further options. In the figure the symbol “!” indicates negation, the states with a label followed by a colon occur in the indicated arrangement and the states outlined in dashes are extensions to this diagram currently operating as direct connections to the next labeled state in the diagram.

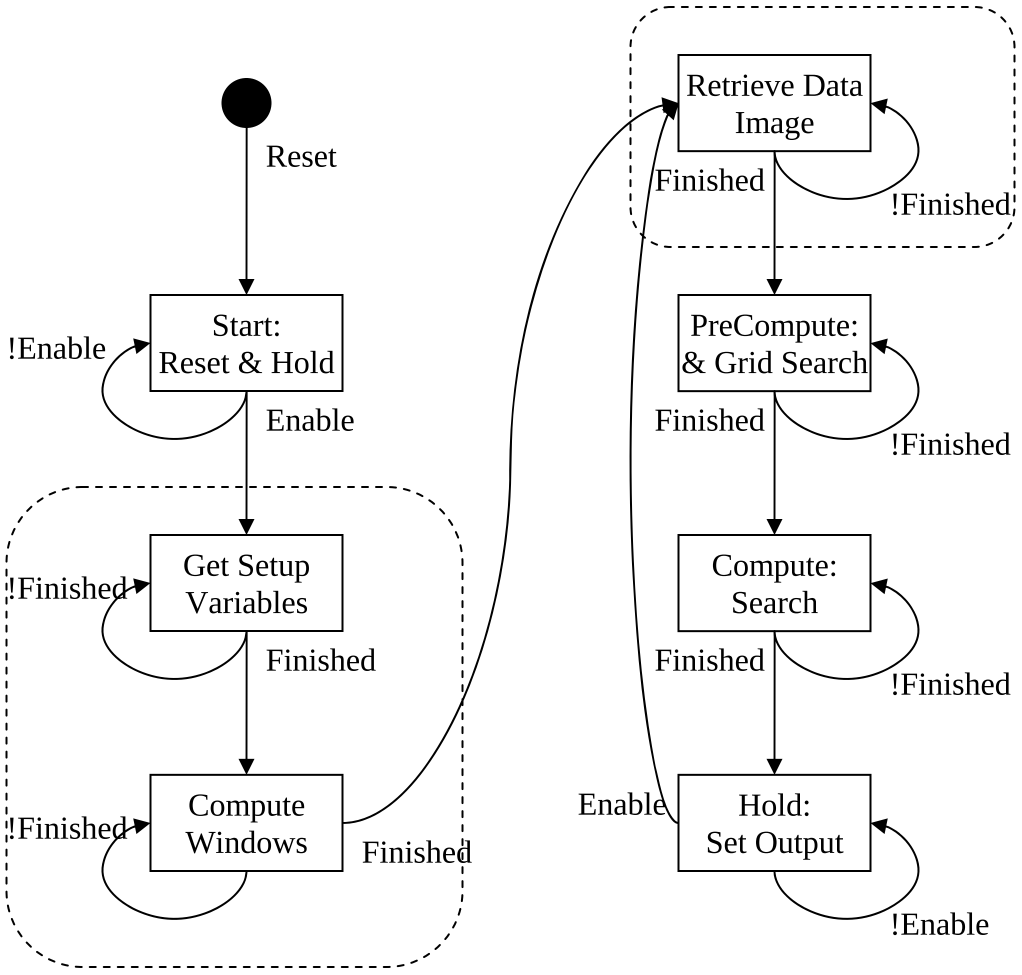


Figure 22. Moore State-Machine for Implementation

The implementation uses 32 bit integer rather than floating-point computations to increase speed and reduce complexity; therefore, to preserve accuracy of results the pixel values of the true image and the log of such values simply scale up by the required accuracy, typically four decimal places for the simulated light-levels, allowing correct computation of the log-likelihood. The final output is also an integer value, which is a linear index into the up-sampled true image vector and the estimated shift computes directly from Equation 61.

$$Shift = \frac{Result[-1] - \frac{IvuL - 1}{2}}{\frac{N}{S}} \tag{61}$$

Shift, Estimated Shift Value (pixels)

Result, VHDL Module Result (array Position)

IvuL, Up-Sampled True Image Length (photons)

S, Amount Multiplied by Nyquist for Over-Sampling (unitless)

[-1], Optional Subtraction for Indexing Starting at One

The PreCompute and Compute states both provide loop control for searching over the log-likelihood values, and currently perform tasks in a clocked-combinatorial fashion; however, these nodes could control additional state machines for further robust operation. Not shown in the diagram is a separate module designed specifically to compute the log-likelihood values providing its own loop control to step through the windowed images and perform computations. This does not implement the interpolation or logarithmic portion of the maximum-likelihood optimized algorithm; however, preliminary fitting and timing results indicate that the entire algorithm requires less than four percent of the logic blocks available in the FPGA, while operating at a conservative clock frequency of 33 MHz. This design performs a single log-likelihood computation every twenty clock cycles, which drives a total computation time of less than 13.4 μs for a single estimate on a 32 pixel image allowing for multiplexing of a single estimation algorithm for seventy-five sensors at an update rate of 1,000 Hz. This speed analysis indicates more than adequate temporal efficiency and basic simulation results appear in the next Chapter, while the VHDL code implementing this phase of the algorithm is in Section 4.2.3.

4.3.3. Extensions to Implementation

Including the interpolation and logarithm in the FPGA would simplify the system greatly by allowing a single information bus from the CCD to the FPGA, reducing the number of pins required and lowering overhead. Additionally, converting the loop control statements within the PreCompute and Compute states to small state machines would further reduce the complexity of the VHDL code, providing extensibility and versatility to the system.

4.4. Summary

This Chapter provides the analyses for three major investigations in tracking and wave-front sensing; development of a CRLB for any tracking or wave-front estimation techniques, optimization of the known vector-projection maximum-likelihood algorithm, and hardware implementation. To validate the new approaches and ensure actual results match the expectations given by analysis and theory, numerous simulations and parameterizations in the next chapter cover the performance of operation in different environments.

V. Results and Discussion

This Chapter brings together the previous work to illustrate in an intuitive fashion the performance, feasibility, and completeness of the research and analysis performed. First presented are the detailed results of simulation for the two-dimensional Gaussian, or LGS, which serve to narrow the range of interest before displaying aggregate results for the tracking and other wave-front sensing applications. Also included are the synthesis and simulation results for the VHDL implementation of the maximum-likelihood optimized algorithm. As indicated in the Chapter on modeling, the simulation step size, 0.1 pixels, is larger than the search step size as set by the CRLB unless indicated different for a particular plot. The total number of noisy trials selected for each realization is 100 trials-per-shift-per-dimension, with the statistics for each dimension averaged together, unless indicated different for a unique data set.

5.1. LGS / 2D Gaussian - Modeling and Simulation Results

The laser guide star is a unique case in that the analysis providing the CRLB uses this type of image for the bound on variance for any estimator, and not only confirms the design choice based on the CRLB but also provides evidence that the bound is indeed a lower bound on variance for wave-front sensing. The parameters used for modeling the Gaussian are a light-level, C, of 300 photons and a standard deviation, σ_i, of two pixels as this is the approximate width of a diffraction limited PSF.

5.1.1. Effects of Image Projection

To provide estimates for two-dimensions, projection of the Gaussian image, and any image, requires either an equally divided light-level or an equally divided time interval, both of which can result in a reduced light-level and reduced SNR. Figure 23 and Figure 24 illustrate the effects of dividing the light between two sensors using the whole image for the Shack-Hartmann wave-front sensor, or cent corresponding to the first CRLB, and projected images at half intensity for the SWAT wave-front sensor corresponding to the second CRLB. For small window sizes, the edge effects not accounted for in the derivative for the CRLB appear by the sensors dipping beneath this bound; however, the larger 16 pixel image illustrates this is a true lower bound for zero-shift estimation and any shifts within approximately six pixels of either window edge. This data represents performance results without a background, *Bg*; and this parameter greatly affects the centroid algorithm as presented later in this chapter in Sub-Section 5.3.3.

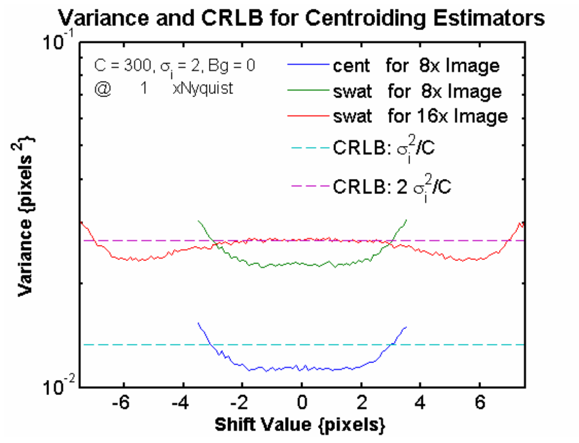


Figure 23. Variance of Centroiding Algorithms and CRLB

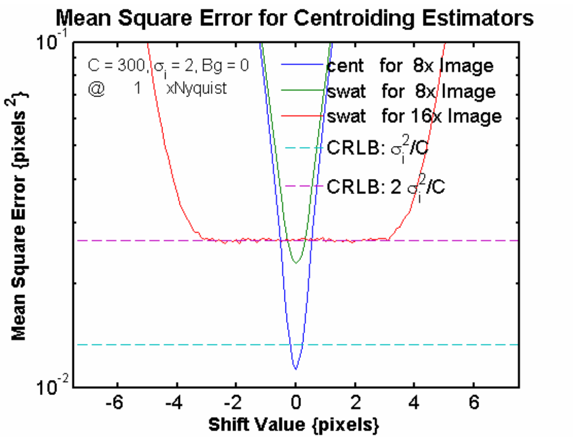


Figure 24. Mean Square Error of Centroiding Algorithms

Since the MSE is the bias squared plus the variance as indicated by Sub-Section 3.5.3, the CRLB is still a lower bound on this statistic as well; therefore, MSE plots also include the CRLB, with both plots computed over 10,000 trails-per-shift in each dimension for this simulation. The MSE indicates that although the variance is half for the Shack-Hartmann wave-front sensor, the bias remains constant regardless of light-level, and this bias overwhelms the noise error, making the estimation performance of the Shack-Hartmann and SWAT sensors equally poor for such a small image size.

Given that the Shack-Hartmann wave-front sensor is not a projection-based estimation technique, it is not feasible to search over plus-or-minus four waves of tilt in a reasonable amount of time. The SWAT wave-front sensor is eight-times faster in reading data for an eight pixel image, which only covers plus-or-minus 3.5 pixels or 1.75 waves of tilt, eliminating the typical Shack-Hartmann centroiding wave-front sensor from further discussion.

5.1.2. Detailed Bias

As mentioned previously, the bias indicates the best-case operation of a sensor and is difficult to remove; therefore, a minimal bias is ideal for any type of sensor. Figure 25 displays the SWAT and maximum-likelihood wave-front sensors for the minimum required image sizes to search plus-and-minus four waves of tilt. This simulation uses a step size of 0.001 pixels to capture the quantization error caused by the CRLB set search step size of approximately 0.008 pixels.

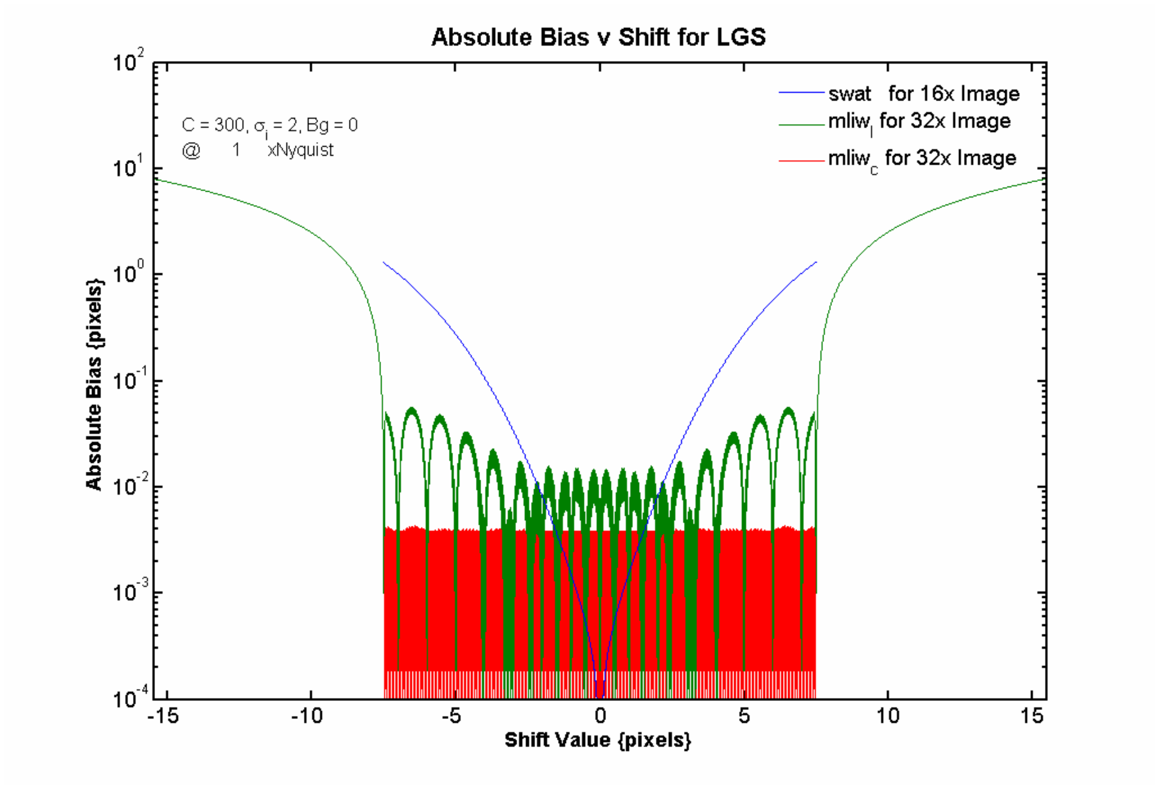


Figure 25. Absolute Bias v Shift for LGS

The most important conclusions to draw stem from the clear improvement in bias of the maximum-likelihood sensor, which windows the data and the true image before shifting the image, over the SWAT wave-front sensor and the nearly order of magnitude difference between the linear, subscript l, and cubic-spline, subscript c, interpolated algorithms.

The expected quantization noise of approximately 0.004 pixels due to the search algorithm's interpolation step size, which the CRLB set, is nearly perfect with the cubic-spline interpolated algorithm. This graph indicates the unbiased nature of the maximum-likelihood algorithm for shifts in the window, provided the search step size is arbitrarily small while making use of an accurate interpolator, further illustrating that the CRLB is a correct lower bound for this model in noise.

5.1.3. Detailed MSE and VAR

Also indicated previously, the variance illustrates the performance of the algorithm in noise, and tuning of the sensor without changing the light statistics cannot reduce or eliminate this error. The mean square error combines the variance with the bias for an overall performance picture as shown in Figure 27, and the variance in Figure 26, both of which have the CRLB with half the light intensity for the projected image sensors and 10,000 trails-per-shift in each dimension.

The key points to derive from these plots include the clear affect of bias shown in the MSE, making some estimated shifts useless for the SWAT sensor, and the apparent noise rejection efficiency of all sensors for relatively small shifts. The linear interpolator's error appears as a small ripple in both the MSE and the VAR, but nearly matches the cubic-spline interpolator with respect to noise rejection.

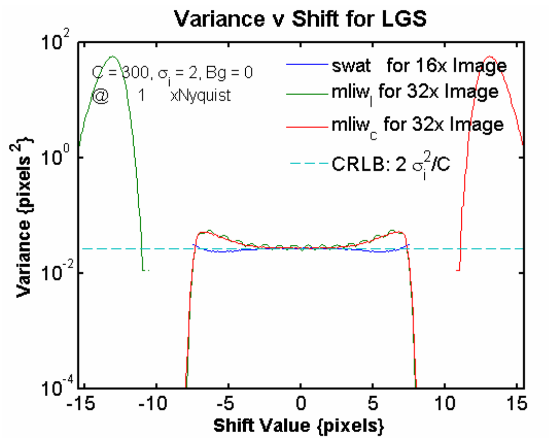


Figure 26. Variance v Shift for LGS

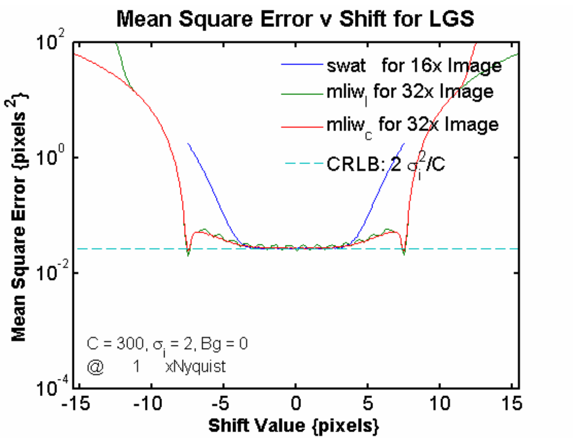


Figure 27. Mean Absolute Error v Shift for LGS

Although it does not affect a wave-front correcting AO system, it is important to note that the maximum-likelihood sensor does not always return the edge of the window for shifts significantly outside of the window, which is a direct result of more than half of the object moving out of the field of view. Since MSE combines the variance and bias of a sensor for a complete picture, further results only display the mean square error.

5.1.4. Image Size Analysis

Image size determines the size of the window and the possibility of the object moving out of the field of view, and although the minimum image sizes are perhaps good enough for wave-front sensing, a search over larger images sizes, as in Figure 28 using 10,000 trials-per-shift-per-dimension, reveals better performance for all sensors. The following results include averages over the previously displayed regions, with a solid line on the graph indicating an average over plus-or-minus one-half wave of tilt while a dot-dashed line indicates an average over plus-or-minus four-waves of tilt or the entire size of the window, whichever is smaller. Included for reference for this plot only is the non-windowing model that requires the functional form of the true image but allows better performance with a smaller image size, serving to illustrate that a regular windowing model approaches this performance with an image 16 pixels larger. The overall statistics indicate an image size of 32 pixels is optimal for the non-windowing maximum-likelihood sensor, mlnw with perfect interpolation and extrapolation using the Gaussian form as indicated by a subscript p. An image size of 48 pixels allows the windowing models that do not require explicit knowledge of the image outside of the window to function with the same performance.

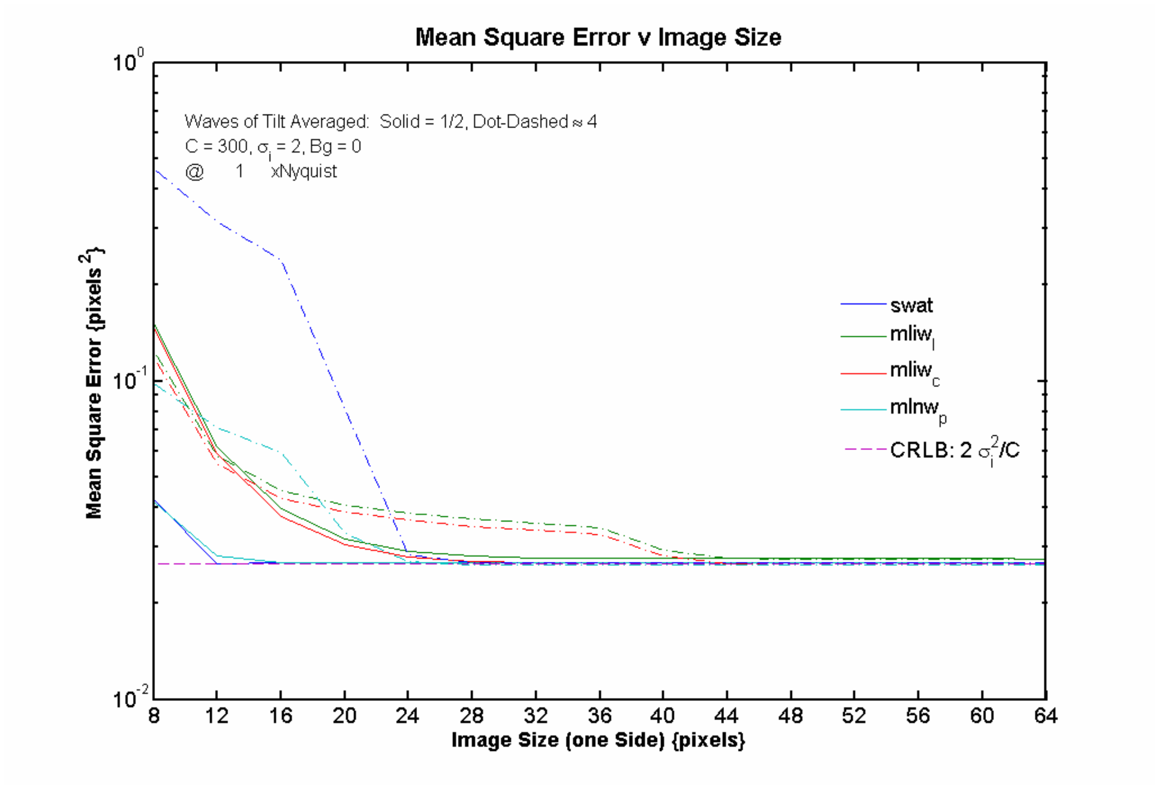


Figure 28. Mean Square Error v Image Size for LGS

These image sizes are the recommended values provided the hardware and subsequent readout speeds are complementary; however, simulation results present the minimum image sizes to illustrate the differences and trends in the sensor models.

5.1.5. Sampling Analysis

Over-sampling of an image aids an interpolator by providing actual data in-between pixels; however, it increases the size of the CCD array and decreases the amount of light per-pixel effectively reducing the SNR; therefore, this is beneficial only if the decrease in bias is greater than the potential increase in variance. The average statistics in Figure 29 indicate that over-sampling aids the linear interpolator somewhat and harms the cubic-spline interpolator to a lesser extent, with the exception of the maximum-likelihood data shifting, mldw, and swat sensors, which perform relatively worse.

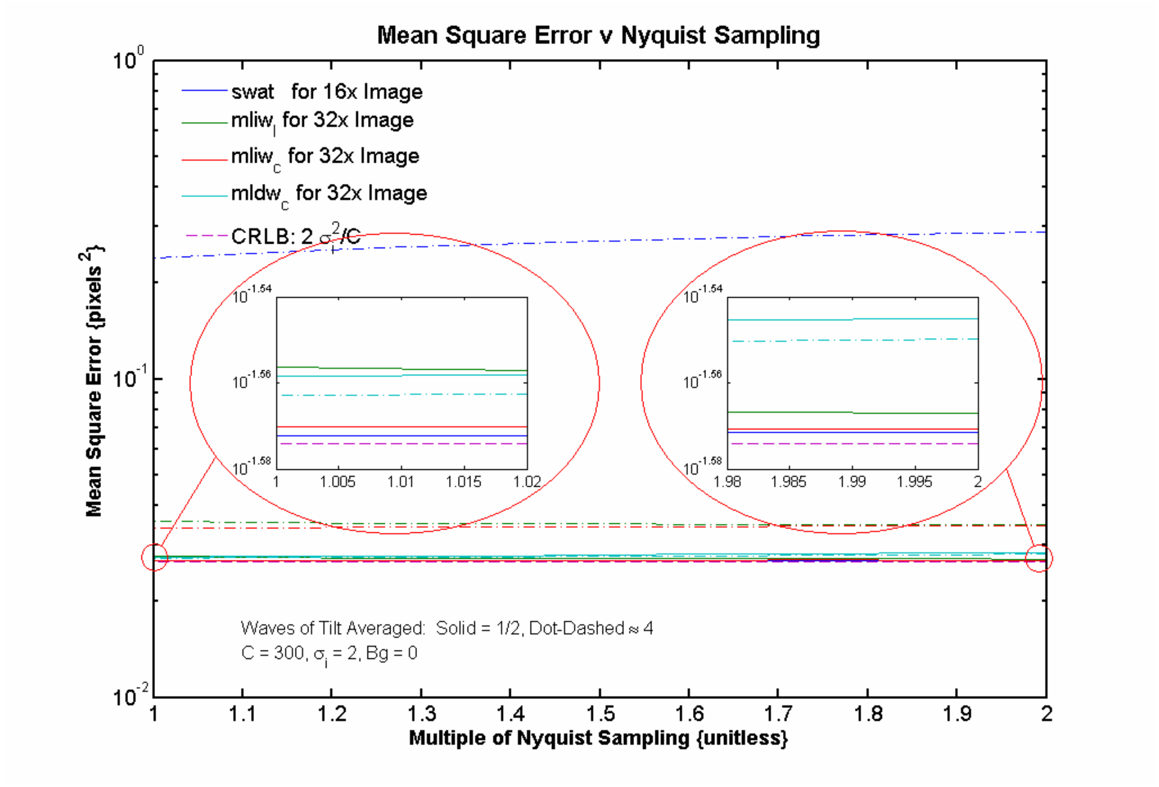


Figure 29. Mean Square Error v Sampling Rate for LGS

Although the corresponding decrease in SNR changes the variance only slightly, the MSE indicates a nearly constant trend in error with the bias driving the results. These results also indicate the best setup for optimal performance of these sensors is sampling at or slightly beyond Nyquist, as the changes in overall error are relatively small.

5.1.6. Background Intensity Analysis

Independent of image size; there can be stray light in the optics setup as well as stray electrons in the CCD array itself, which the sensor sees as a background in the image lowering the overall contrast and decreasing the SNR. The following results in Figure 30 show the importance of minimizing the background for the centroid-based sensor for estimation beyond one-half wave of tilt, and the poor performance of searching by interpolating and shifting the observed data for the maximum-likelihood algorithm.

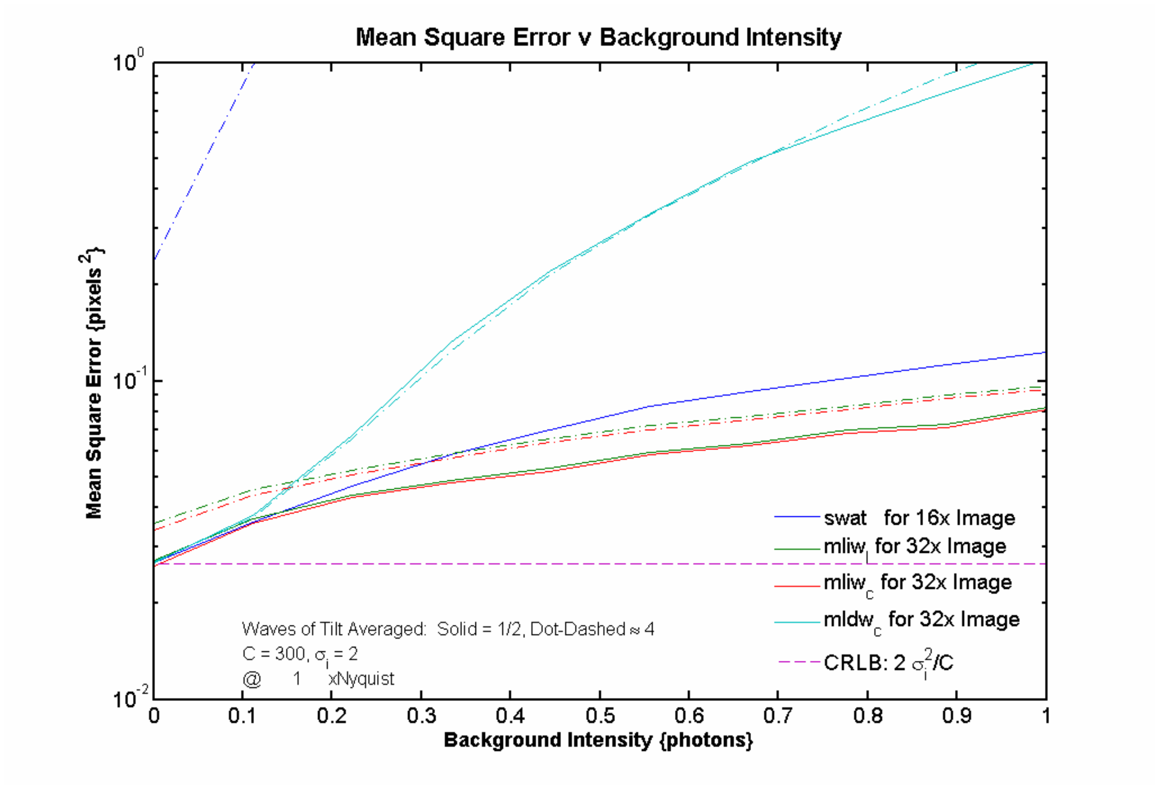


Figure 30. Mean Square Error v Background Intensity for LGS

The windowing maximum-likelihood sensor that estimates the shift by shifting the image has a good tolerance to background, although the growth rate of the MSE for this sensor still appears to be a slow exponential, indicating higher background would disrupt the performance of this sensor significantly as well. As the SWAT and mldw sensors do not perform as well as the typical maximum-likelihood sensor, the discussion will no longer include the centroiding sensor and simulations will not include the mldw sensor.

5.1.7. Light-Level Analysis

The CRLB indicates that an efficient sensor could perform better given higher light-levels, and Figure 31 shows the improved performance relationship for the maximum-likelihood sensor for higher light-levels. As noise levels decrease below the bias of the sensor, the bias drives the remaining error in the model

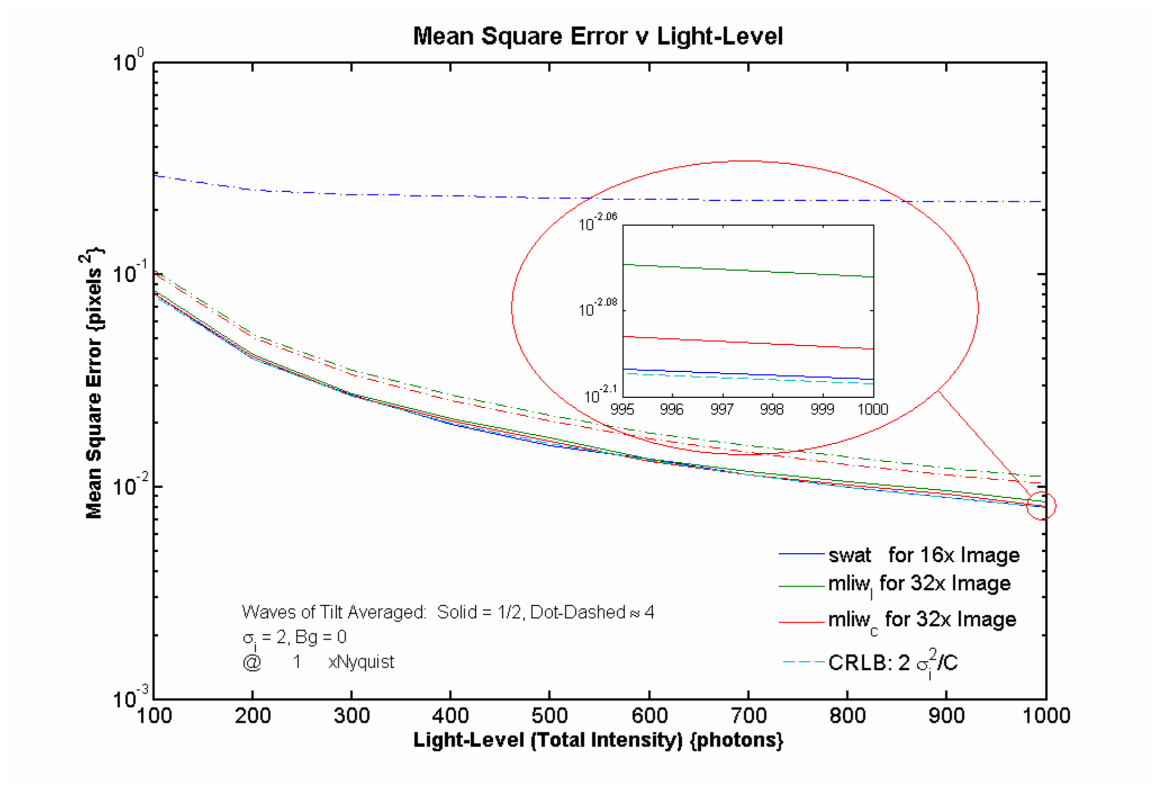


Figure 31. Mean Square Error v Light-Level for LGS

The SWAT wave-front sensor noticeably demonstrates this trend over an average of four waves of tilt. As light levels increase, eventually all models would suffer the same bias-limiting effect; however, the maximum-likelihood sensors have the advantage that a finer search step size driven by the CRLB allows them to remain efficient in noise for higher light-levels.

5.2. Tracking Extended Object - Modeling and Simulation Results

Possibly the most interesting simulation of this chapter is tracking an extended object and these results exercise the extended object model as described in Sub-Section 3.2.2. There are two difficulties with extended objects; first, the object typically fills the field of view, lowering the overall contrast of the image and second, as the object shifts new information enters the field of view, which can cause a sensor's performance to drop and bias to be asymmetric. The contrast ratio of the original Hubble image before proper band-limiting is approximately four-to-one; therefore, these results should suffice for any similar object with a similar telescope OTF and noise statistics. The performance graphs include only the maximum-likelihood sensor as a centroid algorithm cannot cope with new information entering the scene or an object with multiple peaks in the image. Additionally, the dimensions are separate to illustrate the performance for a low-contrast orientation and an orientation containing the full contrast of the object, which indicates that proper tuning of the sensor or optics must occur before tracking is possible.

5.2.1.   Image Size Analysis

As opposed to Figure 30 for the LGS case, Figure 32 illustrates an increase in required image size for acceptable tracking performance.

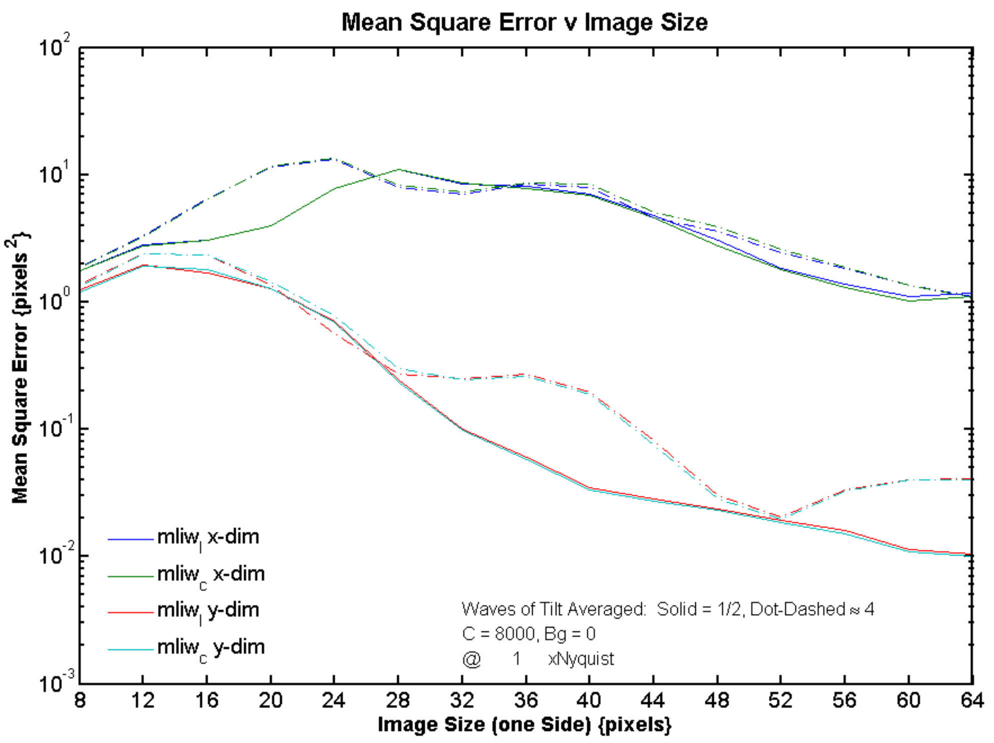


Figure 32. Mean Square Error v Image Size for Hubble Tracking

This graph also indicates that the CRLB derived for the LGS using the same parameters is indeed a lower bound for tracking with this particular image of the Hubble telescope as $2\sigma_i^2 / C = 1\text{x}10^{-3}$-pixels^2. The subsequent simulation results show only the recommended image size of 48-by-48 pixels, except for the sampling analysis to show a complete sampling range.

5.2.2. Sampling Analysis

Figure 33 indicates that over-sampling does not affect the performance of the sensor. Again, the recommendation is to sample at or slightly beyond Nyquist sampling criteria to avoid aliasing of the true image.

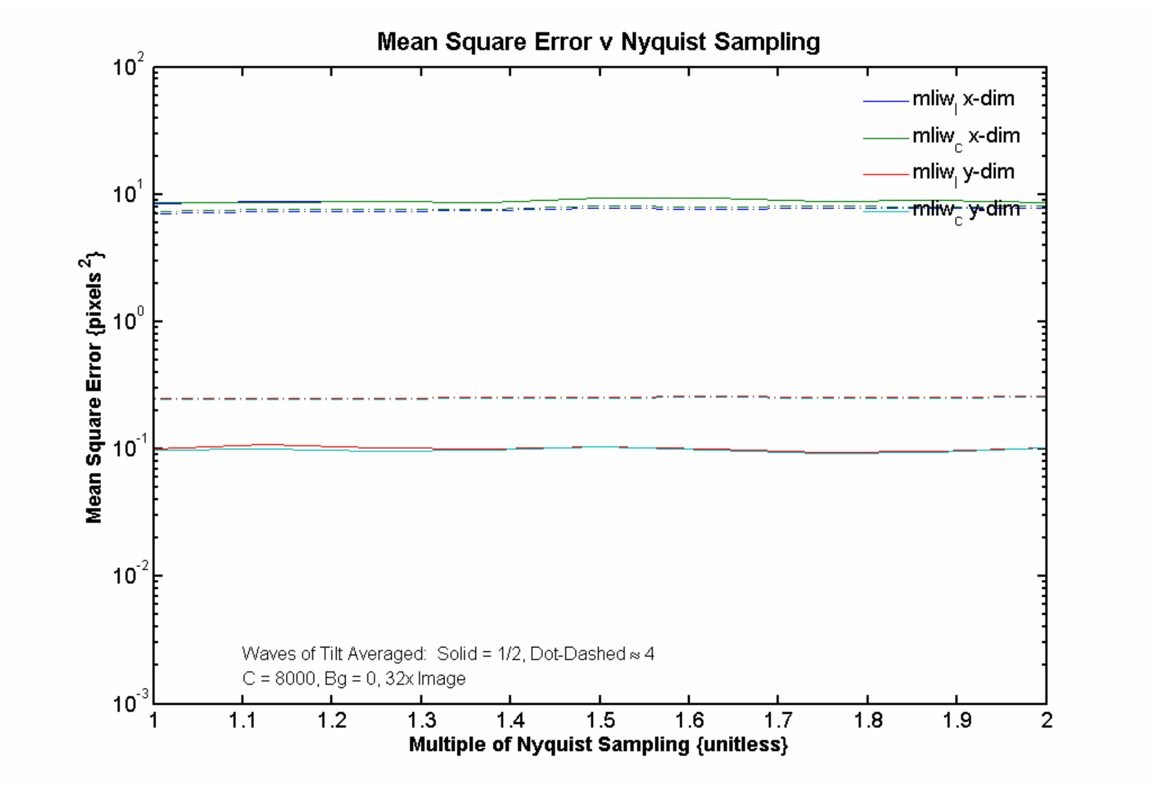


Figure 33. Mean Square Error v Sampling Rate for Hubble Tracking

5.2.3. Background Intensity Analysis

As an extended object already has a background, addition of further background light only reduces the contrast and produces small changes overall in the performance of the maximum-likelihood sensors using either type of interpolation. Figure 34 illustrates the reasonable tolerance to background in the y-dimension and the same tolerance, but consistently poor performance for the low-contrast x-dimension.

5.2.4. Light-Level / Total Intensity Analysis

Much like the laser guide star, the maximum-likelihood sensor performs better with higher light-levels as shown in Figure 35; and the roughness of the curve here is due to the limited number of trails for such a large bias and variance.

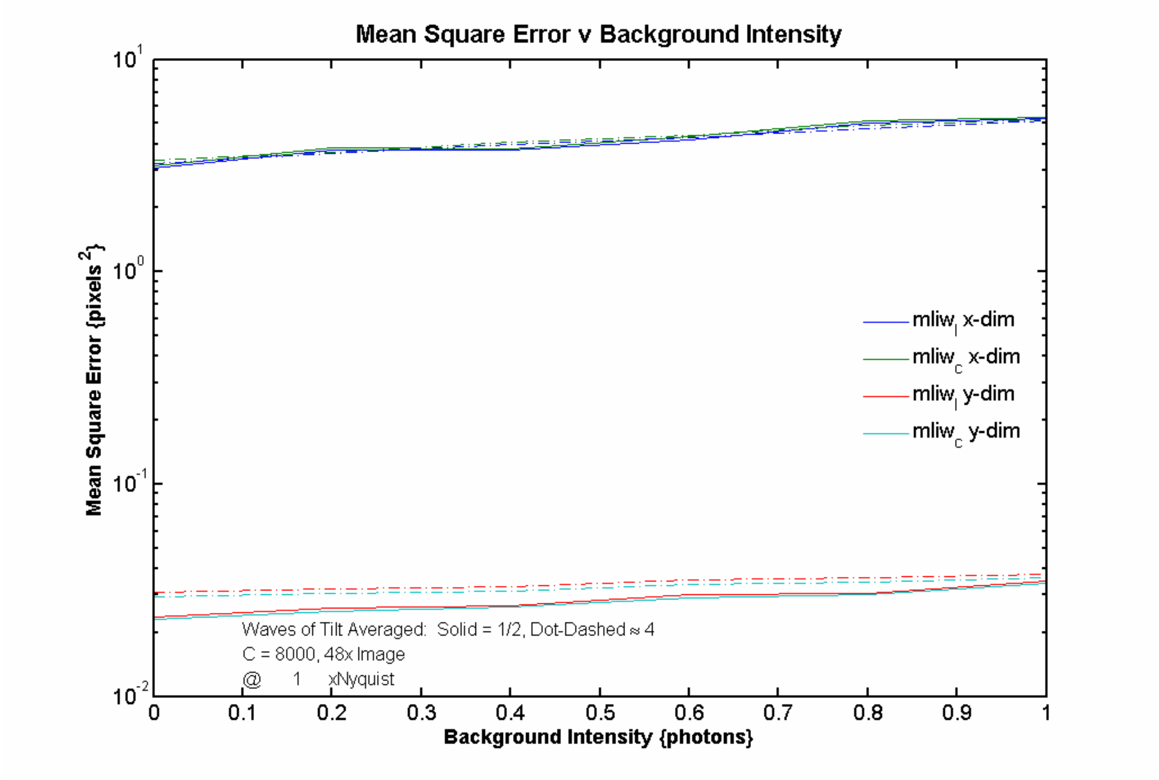


Figure 34. Mean Square Error v Background Intensity for Hubble Tracking

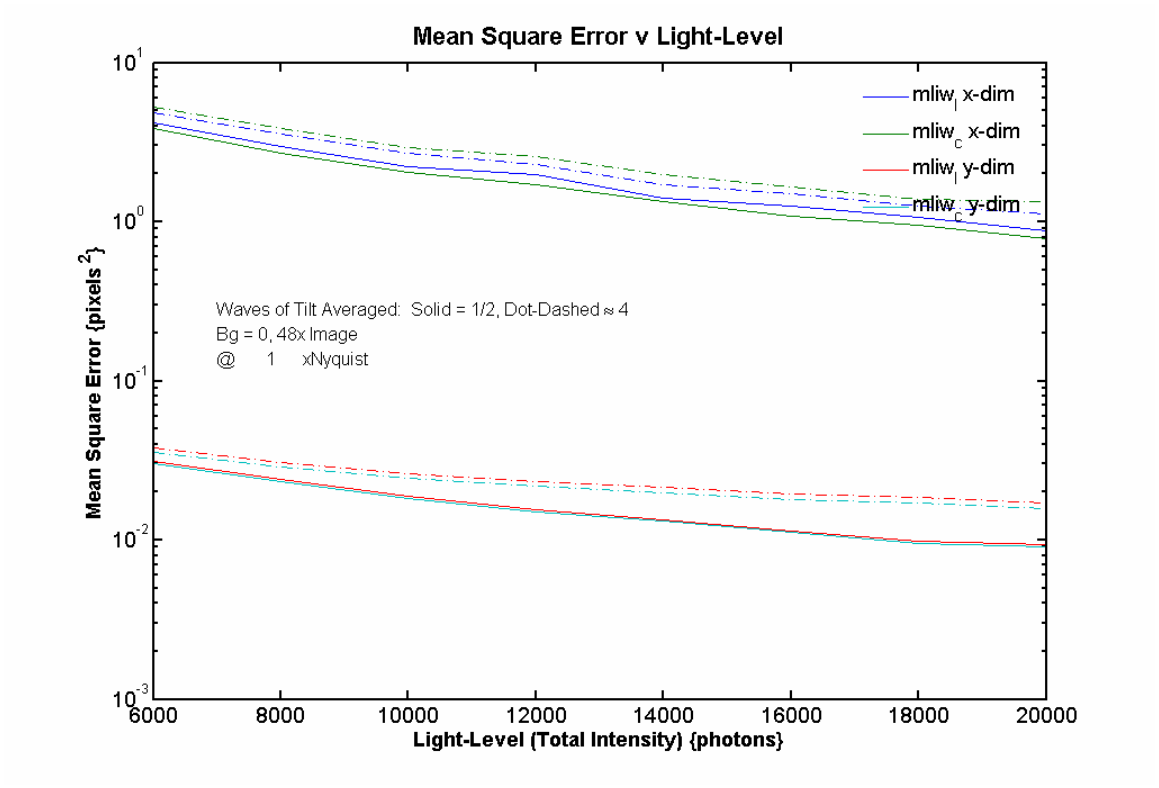


Figure 35. Mean Square Error v Light Level for Hubble Tracking

5.3. WFS Extended Object - Modeling and Simulation Results

Adaptive optics systems attempt to correct atmospherically induced wave-front distortions for any object the researcher wishes to view; therefore, a simulation of a wave-front sensing application on the Hubble provides further insight into the utility of this sensor, and the possibility of tracking and imaging with the same telescope and sensor setup.

5.3.1. Image Size Analysis

Figure 36 again confirms that wave-front sensing over plus-or-minus four waves of tilt is possible at the minimum image size, but the recommended image size for the maximum-likelihood wave-front sensor remains a reasonable 48 pixels in length.

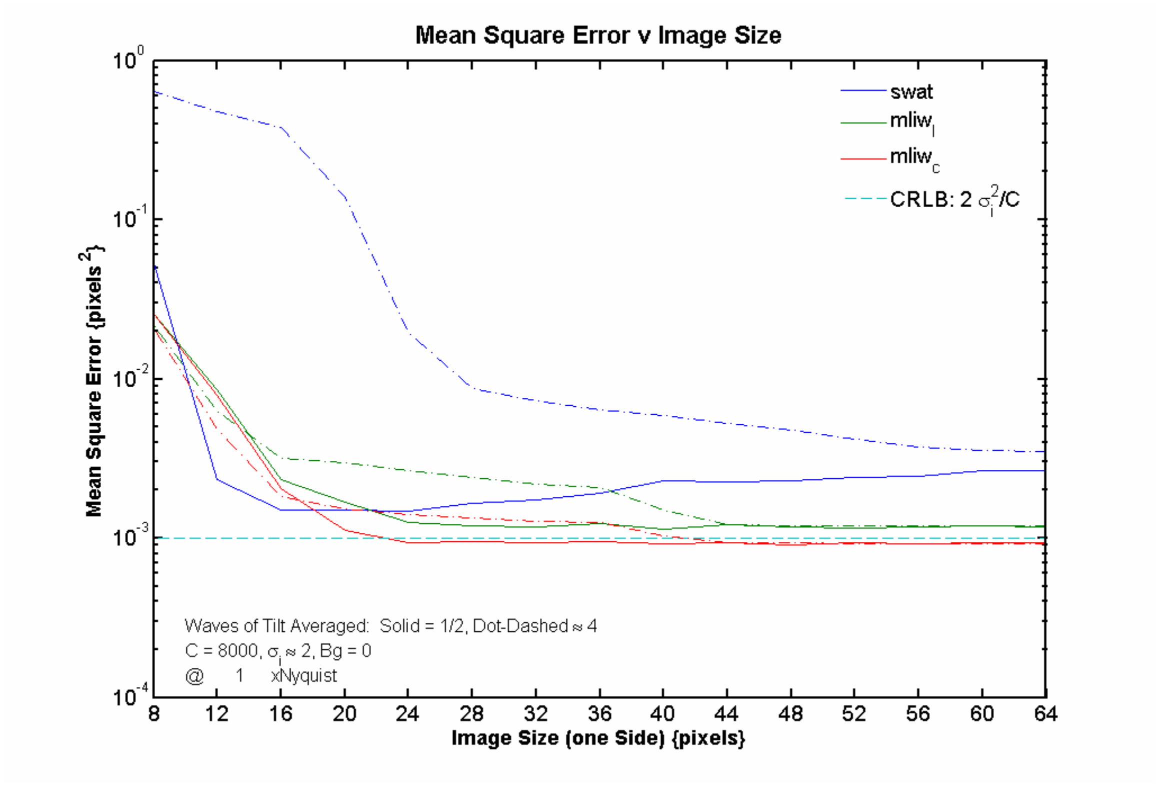


Figure 36. Mean Square Error v Image Size for Hubble WFS

The maximum-likelihood wave-front sensors maintain good performance for larger image sizes, while the centroiding algorithm's performance decreases as the image size increases due to the greater incorporation of the image's natural background. This image also indicates that the CRLB is a good approximation, given the parameter σ_i is an approximation for the average image shape in both dimensions of the image.

5.3.2. Sampling Analysis

This simulation results in Figure 37 indicate the same increase in performance with respect to over-sampling for the linear interpolated sensor, and the same slight decrease in performance for the centroiding and cubic-spline interpolated sensor.

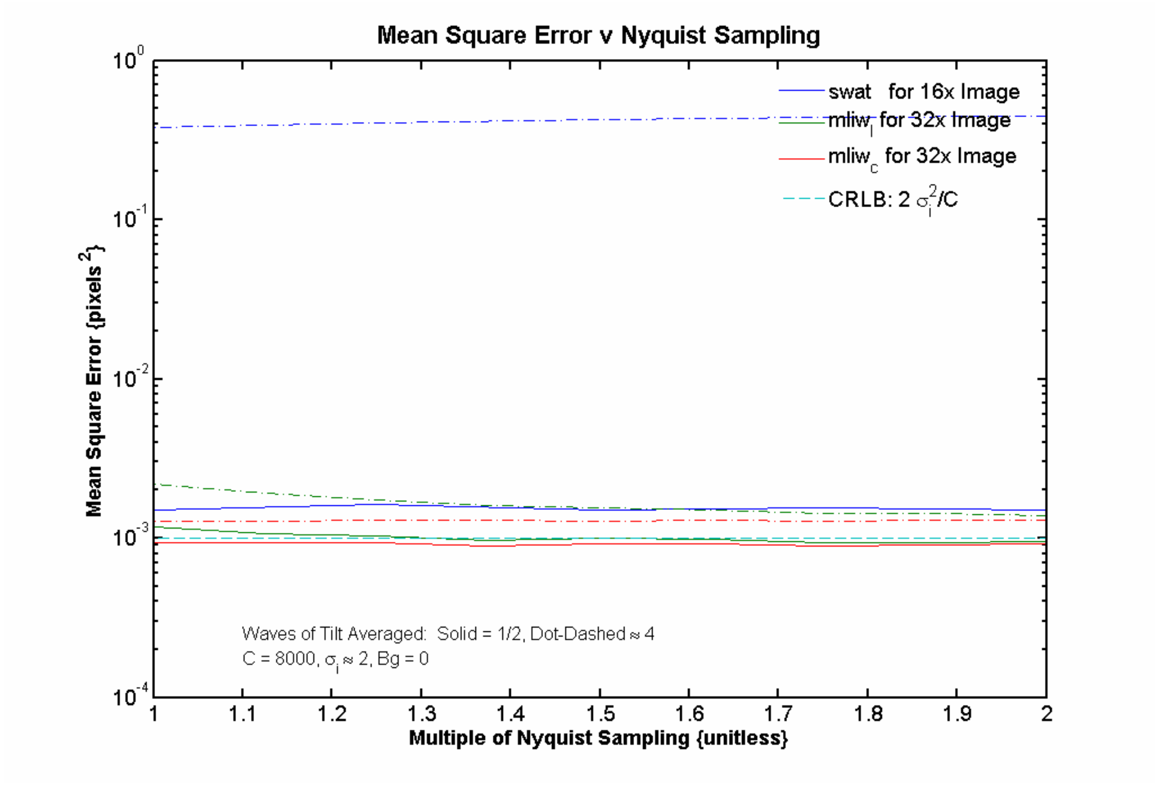


Figure 37. Mean Square Error v Sampling Rate for Hubble WFS

5.3.3. Background Intensity Analysis

Despite the inherent background of the Hubble image, the maximum-likelihood sensor continues to demonstrate strong tolerance to further background light in Figure 38, with very reasonable mean squared error performance. The swat wave-front sensor degrades at a slower rate than seen with the laser guide star due to the higher light level of this object.

5.3.4. Light-Level / Total Intensity Analysis

As the CRLB predicts, Figure 39 indicates all sensors benefit from the increased light-levels; however, the increased light-level appears to aid the cubic-spline interpolator slightly more than the linear interpolator due to the linear interpolator's added error.

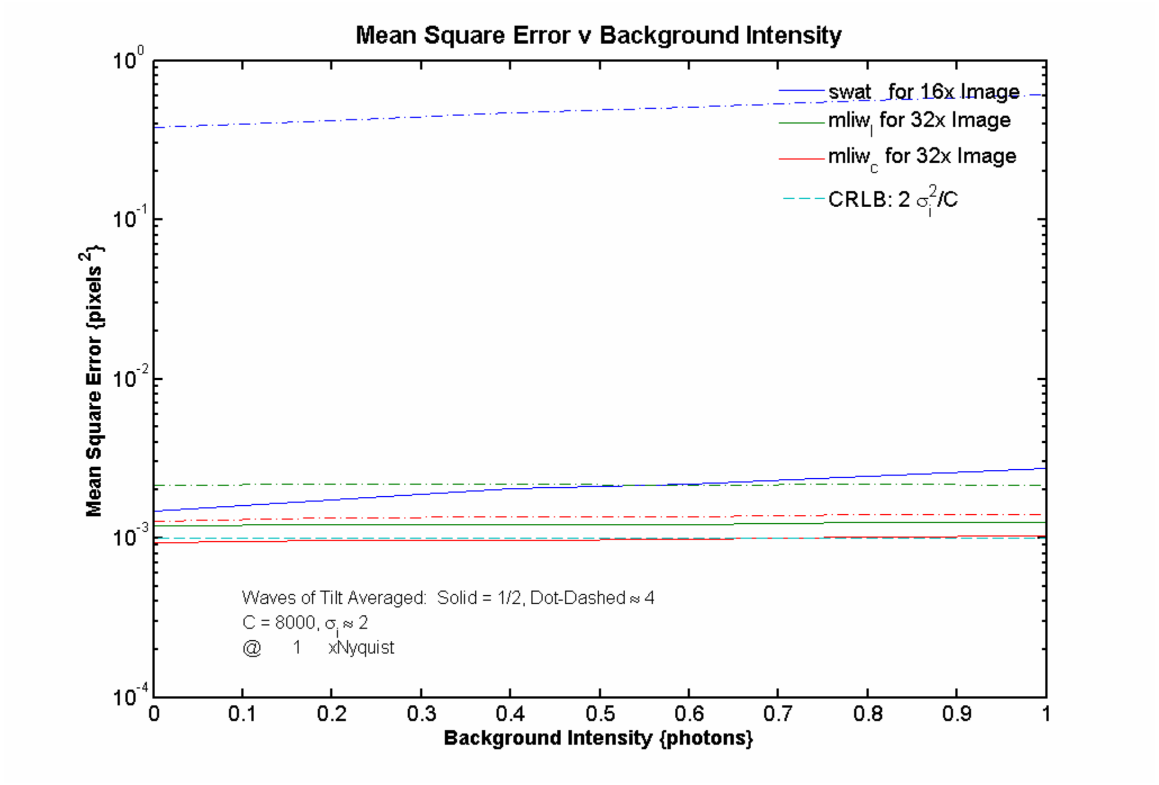


Figure 38. Mean Square Error v Background Intensity for Hubble WFS

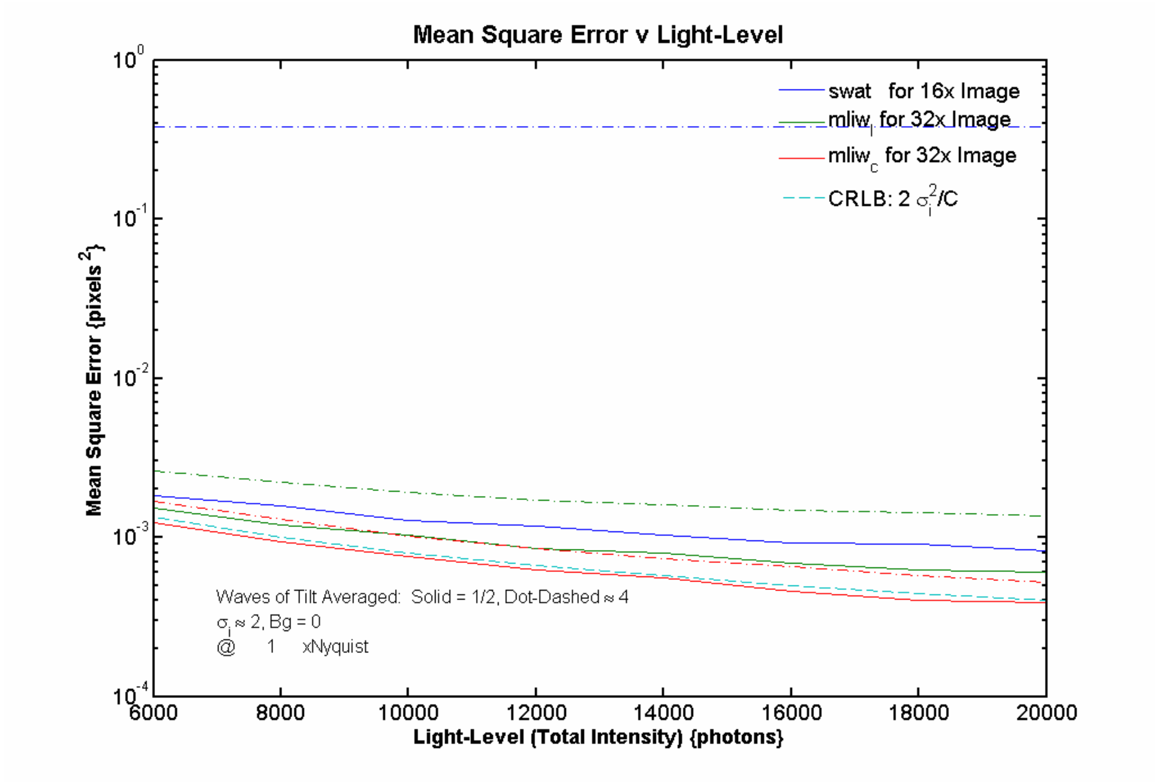


Figure 39. Mean Square Error v Light-Level for Hubble WFS

These simulations illustrate excellent performance for the vector-projection maximum-likelihood wave-front sensor as well as validate the sensor is efficient with respect to the CRLB for larger image sizes. The CRLB also holds for other image types, given an appropriate estimate of σ_i.

5.4. Implementation / VHDL Simulation Results

The simulation results for VHDL represent the estimated speed of operation for the hardware implementation of the sensor; however, these results do not fully test the implementation and further validation should proceed before reliance on this realization of the sensor.

5.4.1. Targeted Device Resource Summary

The following results in Table 7 and Table 8 indicate the usage of the logic elements in the FPGA and the maximum clock speed that the device could operate using the Cyclone II EP2C70F89618 processor.

Table 7. Resources Required for Synthesis

Resource	Number Used	Number Available	Percent Used
Logic Elements	2,447	68,416	< 4%
Registers	491		
I/O Pins	228	622	37 %
Memory Bits	0	1,152,000	0 %
Embedded 9 bit Multipliers	6	300	2 %

Table 8. Preliminary Timing Analysis

Timing Type	Value
Clock	53.79 MHz (18.59 ns)
Worst-Case Setup Time (t_{su})	23.381 ns
Worst-Case Hold Time (t_h)	-1.225 ns
Worst-Case Clock to Output (t_{co})	14.21 ns

Additionally, the preliminary estimated power consumption is a low 260 mW allowing for reduced cooling requirements in an embedded environment. These results indicate that a clock period of 30 ns is a safe value for setup and hold times while maintaining the performance of the estimation algorithm, which results in a total compute time of less than 13.4 μs, allowing for time-division multiplexing of one module for up to 75 different wave-front sensors at an update rate of 1,000 Hz.

5.4.2. Test Bench Simulation Results

This simulation serves to indicate the total time required to complete one log-likelihood search for a 32-by-32 pixel image, and does not indicate complete accuracy or validate the implementation outside of the simple, non-realistic test case presented, as it has only 44.8 % test coverage. Figure 40 indicates the form of an individual log-likelihood computation and is a zoomed view of the beginning of Figure 41, which shows the overall computation for a nearly worst case of 23 log-likelihood computations.

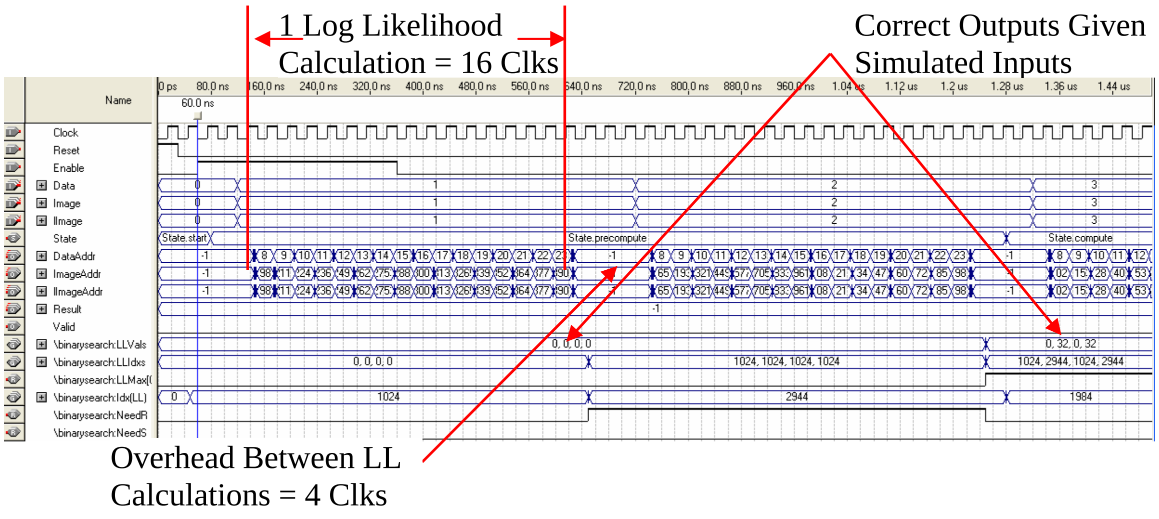


Figure 40. Zoomed in View of First Portion of VHDL Simulation

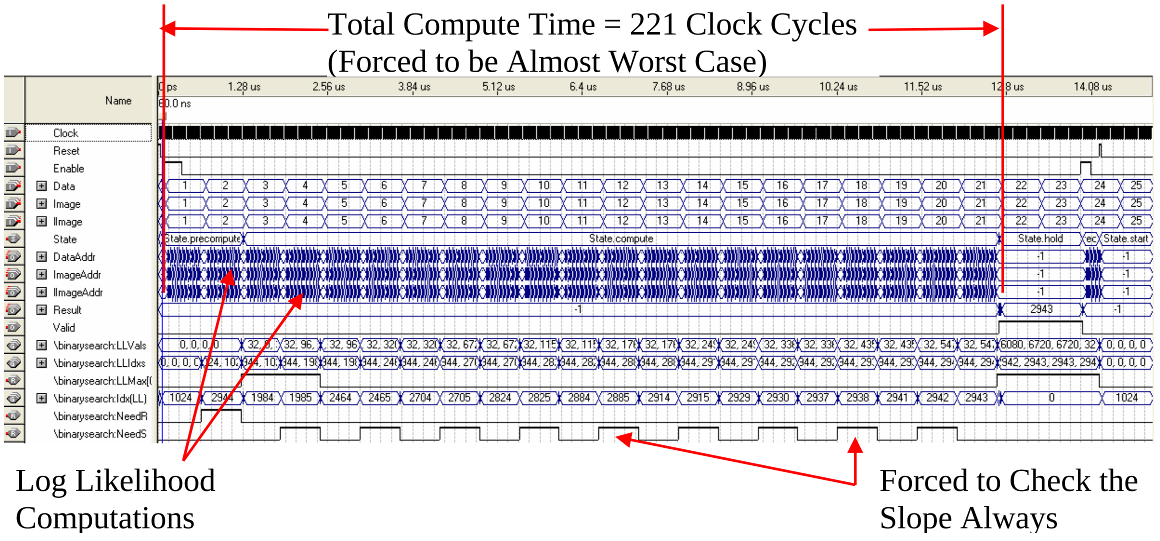


Figure 41. Full View of VHDL Simulation

These results indicate the total required operating time as well as proper operation for the simplistic test case presented. Complete algorithmic flow testing and addressing verification are follow-on research areas in a complete implementation.

5.5. Summary

The three main areas of research results presented in this chapter conclude nearly half of the developmental life cycle of a project including theoretical research, algorithmic development, and hardware implementation, with the remaining portion of the life cycle including at a minimum hardware realization, testing, and maintenance. As an additional benefit, further simulation results parameterizing the sensors over a wider range appear in Appendix D. The analysis for the CRLB provides implementation and validation benefits, while the developed log-likelihood search algorithm and implementation provide a solid foundation and proof of concept for implementation.

VI. Conclusions

From the results and discussion, it is clear that the vector-projection maximum-likelihood wave-front and tracking sensor is quite versatile and feasible to implement. This research intended to and succeeded in providing definitive results for characterization and implementation of the maximum-likelihood sensor through the use of applied theory, robust modeling, and sound implementation techniques. Combined with the power of a fast search algorithm, maximum-likelihood could become the new standard in embedded estimation techniques.

6.1. Key Contributions

Not only did this research develop a Cramer-Rao lower bound for any wave-front sensing or tracking application but it further reduced this complex theoretical model to a simple point solution providing an easy method to predict and validate real-life results. The results of the bias analysis provides targeted implementation information as well as the noise statistics which further solidified the utility of this sensor by indicating the greater efficiency compared to the centroiding algorithms currently in use. The optimized search algorithm with noise rejection capability designed for a concave-down log-likelihood is useful for any maximum-likelihood application with a similar likelihood, log-likelihood, or other type of curve to seek a maximum or minimum over. Finally, the development method and implementation of this optimized algorithm in hardware reduces the life-cycle time greatly for current use, while providing a road map for future implementations of other complex search algorithms or embedded software.

6.2. Lessons Learned

Several difficult areas of this research could have been simpler if more time and care had focused on proper modeling of the environment for statistical analysis and simulation. It is impossible to determine if an algorithm or implementation is correct if the information sent to the element of research is not correct, as only unexpected results occur. In addition, reliance on modern simulation tools often led to further problems as the implementation of these tools is not always clear and may provide inaccurate results in limited circumstances. Outside of these areas of difficulty, the research went quite smoothly by capitalizing on the background and understanding provided by a thorough education.

6.3. Further Research

As with any research there are many areas in which improvements, extensions, and developments are available; therefore, this section concludes with the possible follow-on work associated with the areas of research investigated.

The Cramer-Rao lower bound only includes the intensity of light as a parameter, ignoring contrast, which is the single largest contributor to reduced performance for an extended object. The realization of an inclusion or relation to contrast would provide even greater utility to the simplistic yet effective zero-shift minimum variance calculation. Additionally, the CCD estimates each pixel in an image during the capture process; therefore, a more accurate bound would jointly estimate the current parameters with each pixel intensity.

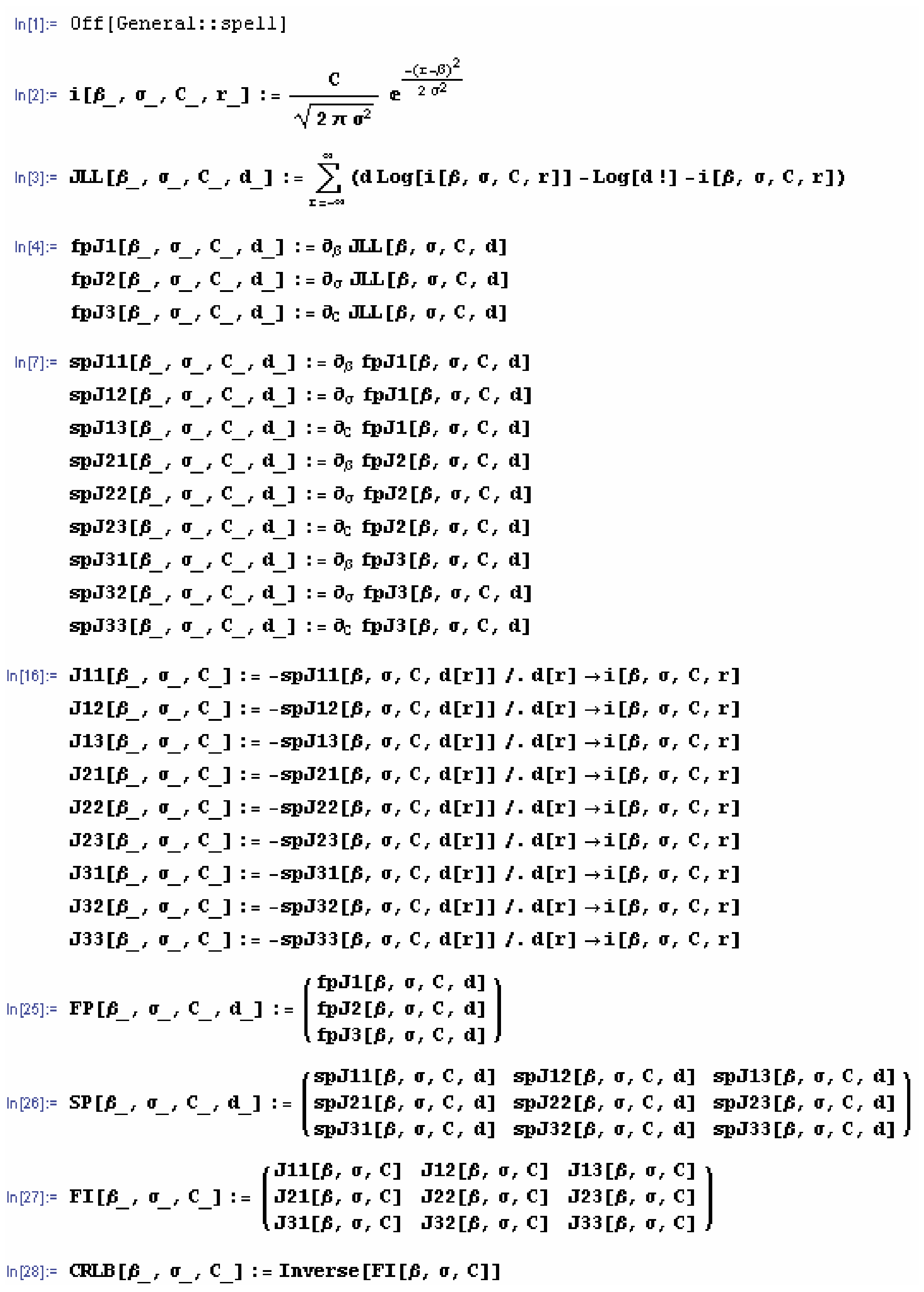
The vector-projection maximum-likelihood tracking and wave-front sensor has numerous areas of investigation, only some of which stem directly from this research. It may be possible to compute a closed-form derivative for any image, allowing for single-step computation for ML estimation. With the ability to dynamically search the log-likelihood, it would be very possible to perform on-the-fly interpolation and possibly even automatic interpolation level adjustment based on the results of the optimized search algorithm. Another important area of investigation is the use of this search algorithm for joint estimation in two-dimensions, either using vector-projection or the entire image, or other techniques such as phase diversity estimation. An enhancement to the tracking application would be the inclusion of automatic light-level normalization between estimations, which would provide greater accuracy and better performance.

The embedded implementation should match the performance of a simulation algorithm; however, further state-machine analysis would simplify the hardware layout and provide a more robust solution. Further testing and development of this algorithm to validate the design could allow for immediate implementation. Finally, implementation of the required interpolation and logarithmic functions would increase the overall productivity of an embedded system and simplify the data transfer requirements between different systems greatly.

Even without the above extensions, maximum-likelihood has a promising future in the arenas of tracking and wave-front sensing applications.

Appendix A: Mathematica Verification of CRLB

A.1. Setup of Formulas



In[1]:= `Off[General::spell]`

In[2]:= $$i[\beta\_, \sigma\_, C\_, r\_] := \frac{C}{\sqrt{2\,\pi\,\sigma^2}}\, e^{\frac{-(r-\beta)^2}{2\,\sigma^2}}$$

In[3]:= $$JLL[\beta\_, \sigma\_, C\_, d\_] := \sum_{r=-\infty}^{\infty} (d\,Log[i[\beta, \sigma, C, r]] - Log[d\,!] - i[\beta, \sigma, C, r])$$

In[4]:= fpJ1[β_, σ_, C_, d_] := ∂_β JLL[β, σ, C, d]
fpJ2[β_, σ_, C_, d_] := ∂_σ JLL[β, σ, C, d]
fpJ3[β_, σ_, C_, d_] := ∂_C JLL[β, σ, C, d]

In[7]:= spJ11[β_, σ_, C_, d_] := ∂_β fpJ1[β, σ, C, d]
spJ12[β_, σ_, C_, d_] := ∂_σ fpJ1[β, σ, C, d]
spJ13[β_, σ_, C_, d_] := ∂_C fpJ1[β, σ, C, d]
spJ21[β_, σ_, C_, d_] := ∂_β fpJ2[β, σ, C, d]
spJ22[β_, σ_, C_, d_] := ∂_σ fpJ2[β, σ, C, d]
spJ23[β_, σ_, C_, d_] := ∂_C fpJ2[β, σ, C, d]
spJ31[β_, σ_, C_, d_] := ∂_β fpJ3[β, σ, C, d]
spJ32[β_, σ_, C_, d_] := ∂_σ fpJ3[β, σ, C, d]
spJ33[β_, σ_, C_, d_] := ∂_C fpJ3[β, σ, C, d]

```
In[16]:= J11[β_, σ_, C_] := -spJ11[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J12[β_, σ_, C_] := -spJ12[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J13[β_, σ_, C_] := -spJ13[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J21[β_, σ_, C_] := -spJ21[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J22[β_, σ_, C_] := -spJ22[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J23[β_, σ_, C_] := -spJ23[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J31[β_, σ_, C_] := -spJ31[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J32[β_, σ_, C_] := -spJ32[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
         J33[β_, σ_, C_] := -spJ33[β, σ, C, d[r]] /. d[r] → i[β, σ, C, r]
```

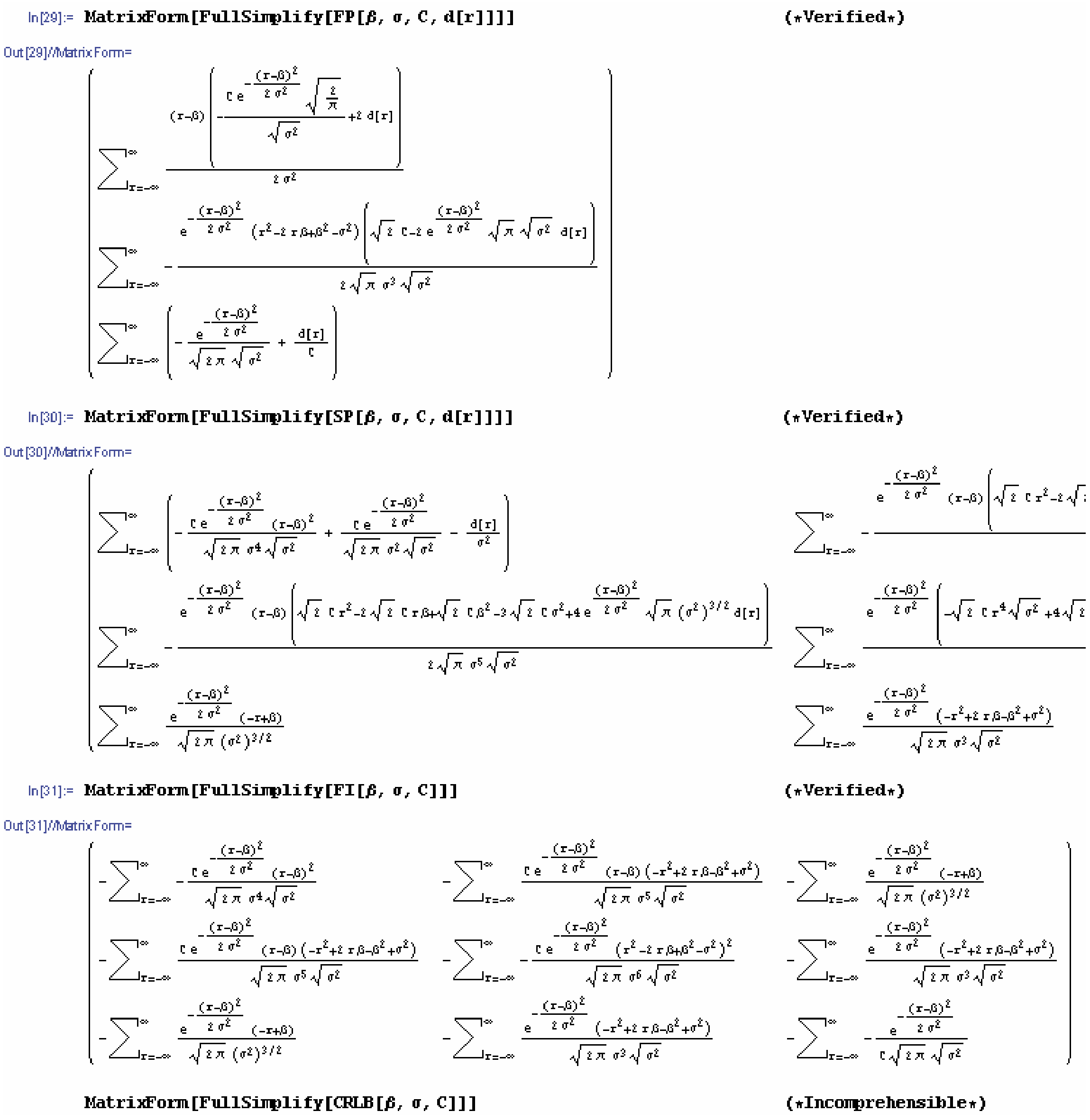
In[25]:= $$FP[\beta\_, \sigma\_, C\_, d\_] := \begin{pmatrix} fpJ1[\beta, \sigma, C, d] \\ fpJ2[\beta, \sigma, C, d] \\ fpJ3[\beta, \sigma, C, d] \end{pmatrix}$$

In[26]:= $$SP[\beta\_, \sigma\_, C\_, d\_] := \begin{pmatrix} spJ11[\beta, \sigma, C, d] & spJ12[\beta, \sigma, C, d] & spJ13[\beta, \sigma, C, d] \\ spJ21[\beta, \sigma, C, d] & spJ22[\beta, \sigma, C, d] & spJ23[\beta, \sigma, C, d] \\ spJ31[\beta, \sigma, C, d] & spJ32[\beta, \sigma, C, d] & spJ33[\beta, \sigma, C, d] \end{pmatrix}$$

In[27]:= $$FI[\beta\_, \sigma\_, C\_] := \begin{pmatrix} J11[\beta, \sigma, C] & J12[\beta, \sigma, C] & J13[\beta, \sigma, C] \\ J21[\beta, \sigma, C] & J22[\beta, \sigma, C] & J23[\beta, \sigma, C] \\ J31[\beta, \sigma, C] & J32[\beta, \sigma, C] & J33[\beta, \sigma, C] \end{pmatrix}$$

In[28]:= `CRLB[β_, σ_, C_] := Inverse[FI[β, σ, C]]`

A.2. Computation of CRLB



```
In[29]:= MatrixForm[FullSimplify[FP[β, σ, C, d[r]]]]                    (*Verified*)
```

Out[29]//MatrixForm=

$$\begin{pmatrix} \sum_{r=-\infty}^{\infty} \frac{(r-\beta)\left(-\frac{C e^{-\frac{(r-\beta)^2}{2\sigma^2}}\sqrt{\frac{2}{\pi}}}{\sqrt{\sigma^2}}+2\,d[r]\right)}{2\sigma^2} \\ \sum_{r=-\infty}^{\infty} -\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}\left(r^2-2r\beta+\beta^2-\sigma^2\right)\left(\sqrt{2}\,C-2e^{\frac{(r-\beta)^2}{2\sigma^2}}\sqrt{\pi}\sqrt{\sigma^2}\,d[r]\right)}{2\sqrt{\pi}\,\sigma^3\sqrt{\sigma^2}} \\ \sum_{r=-\infty}^{\infty}\left(-\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}}{\sqrt{2\pi}\sqrt{\sigma^2}}+\frac{d[r]}{C}\right) \end{pmatrix}$$

```
In[30]:= MatrixForm[FullSimplify[SP[β, σ, C, d[r]]]]                    (*Verified*)
```

Out[30]//MatrixForm=

$$\begin{pmatrix} \sum_{r=-\infty}^{\infty}\left(-\frac{C e^{-\frac{(r-\beta)^2}{2\sigma^2}}(r-\beta)^2}{\sqrt{2\pi}\,\sigma^4\sqrt{\sigma^2}}+\frac{C e^{-\frac{(r-\beta)^2}{2\sigma^2}}}{\sqrt{2\pi}\,\sigma^2\sqrt{\sigma^2}}-\frac{d[r]}{\sigma^2}\right) & \sum_{r=-\infty}^{\infty} - e^{-\frac{(r-\beta)^2}{2\sigma^2}}(r-\beta)\left(\sqrt{2}\,C\,r^2-2\right. \\ \sum_{r=-\infty}^{\infty} -\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}(r-\beta)\left(\sqrt{2}\,C r^2-2\sqrt{2}\,C r\beta+\sqrt{2}\,C\beta^2-3\sqrt{2}\,C\sigma^2+4e^{\frac{(r-\beta)^2}{2\sigma^2}}\sqrt{\pi}\left(\sigma^2\right)^{3/2}d[r]\right)}{2\sqrt{\pi}\,\sigma^5\sqrt{\sigma^2}} & \sum_{r=-\infty}^{\infty} e^{-\frac{(r-\beta)^2}{2\sigma^2}}\left(-\sqrt{2}\,C\,r^4\sqrt{\sigma^2}+4\sqrt{2}\right. \\ \sum_{r=-\infty}^{\infty}\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}(-r+\beta)}{\sqrt{2\pi}\left(\sigma^2\right)^{3/2}} & \sum_{r=-\infty}^{\infty}\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}\left(-r^2+2r\beta-\beta^2+\sigma^2\right)}{\sqrt{2\pi}\,\sigma^3\sqrt{\sigma^2}} \end{pmatrix}$$

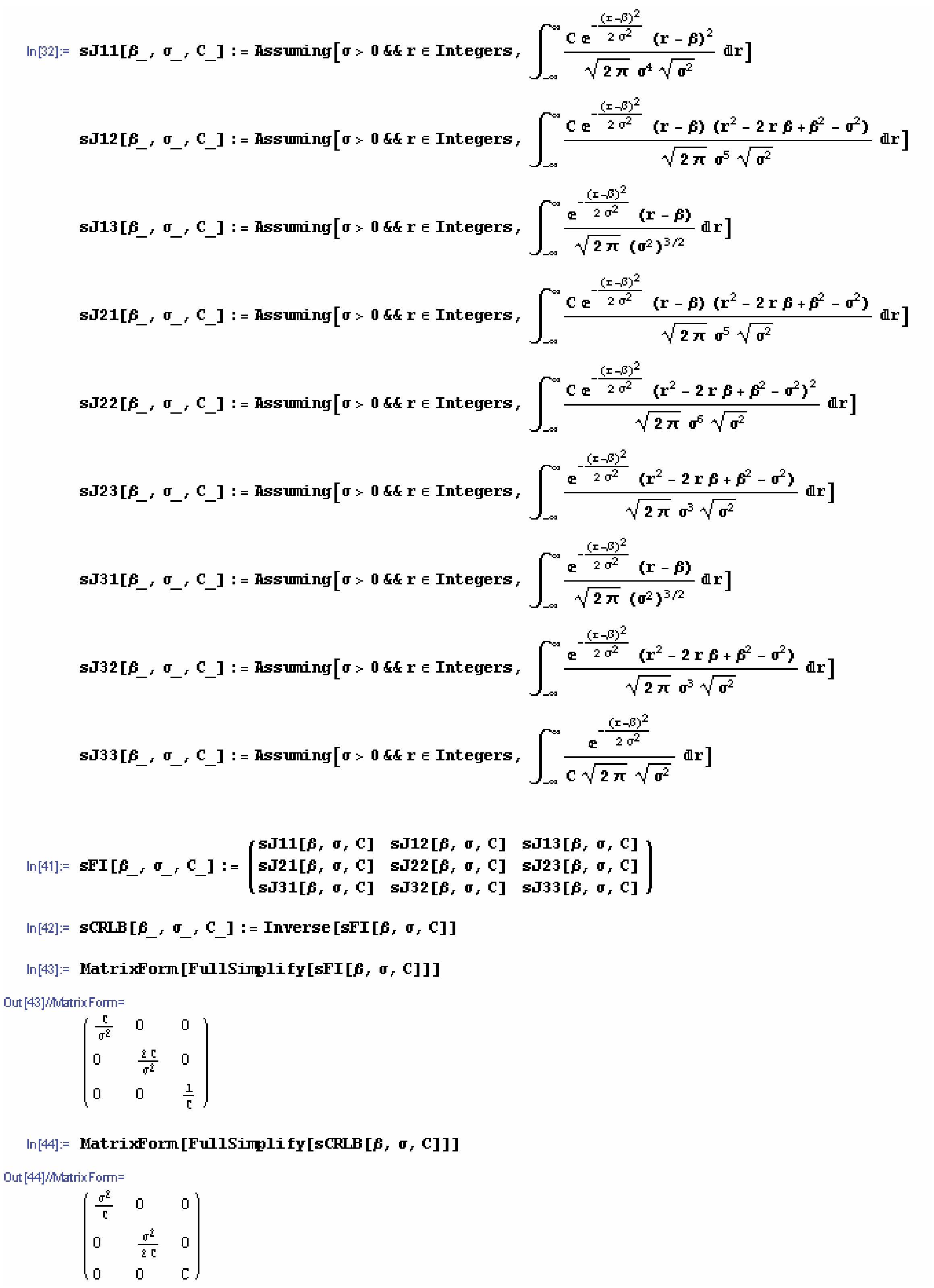
```
In[31]:= MatrixForm[FullSimplify[FI[β, σ, C]]]                    (*Verified*)
```

Out[31]//MatrixForm=

$$\begin{pmatrix} -\sum_{r=-\infty}^{\infty} -\frac{C e^{-\frac{(r-\beta)^2}{2\sigma^2}}(r-\beta)^2}{\sqrt{2\pi}\,\sigma^4\sqrt{\sigma^2}} & -\sum_{r=-\infty}^{\infty}\frac{C e^{-\frac{(r-\beta)^2}{2\sigma^2}}(r-\beta)\left(-r^2+2r\beta-\beta^2+\sigma^2\right)}{\sqrt{2\pi}\,\sigma^5\sqrt{\sigma^2}} & -\sum_{r=-\infty}^{\infty}\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}(-r+\beta)}{\sqrt{2\pi}\left(\sigma^2\right)^{3/2}} \\ -\sum_{r=-\infty}^{\infty}\frac{C e^{-\frac{(r-\beta)^2}{2\sigma^2}}(r-\beta)\left(-r^2+2r\beta-\beta^2+\sigma^2\right)}{\sqrt{2\pi}\,\sigma^5\sqrt{\sigma^2}} & -\sum_{r=-\infty}^{\infty} -\frac{C e^{-\frac{(r-\beta)^2}{2\sigma^2}}\left(r^2-2r\beta+\beta^2-\sigma^2\right)^2}{\sqrt{2\pi}\,\sigma^6\sqrt{\sigma^2}} & -\sum_{r=-\infty}^{\infty}\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}\left(-r^2+2r\beta-\beta^2+\sigma^2\right)}{\sqrt{2\pi}\,\sigma^3\sqrt{\sigma^2}} \\ -\sum_{r=-\infty}^{\infty}\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}(-r+\beta)}{\sqrt{2\pi}\left(\sigma^2\right)^{3/2}} & -\sum_{r=-\infty}^{\infty}\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}\left(-r^2+2r\beta-\beta^2+\sigma^2\right)}{\sqrt{2\pi}\,\sigma^3\sqrt{\sigma^2}} & -\sum_{r=-\infty}^{\infty} -\frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}}{C\sqrt{2\pi}\sqrt{\sigma^2}} \end{pmatrix}$$

```
         MatrixForm[FullSimplify[CRLB[β, σ, C]]]                    (*Incomprehensible*)
```

A.3. Simplification Setup and Solution



In[32]:= $\mathtt{sJ11[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{C\, e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r-\beta)^2}{\sqrt{2\pi}\ \sigma^4 \sqrt{\sigma^2}}\, dr\right]$

$\mathtt{sJ12[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{C\, e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r-\beta)\,(r^2 - 2 r \beta + \beta^2 - \sigma^2)}{\sqrt{2\pi}\ \sigma^5 \sqrt{\sigma^2}}\, dr\right]$

$\mathtt{sJ13[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r-\beta)}{\sqrt{2\pi}\ (\sigma^2)^{3/2}}\, dr\right]$

$\mathtt{sJ21[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{C\, e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r-\beta)\,(r^2 - 2 r \beta + \beta^2 - \sigma^2)}{\sqrt{2\pi}\ \sigma^5 \sqrt{\sigma^2}}\, dr\right]$

$\mathtt{sJ22[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{C\, e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r^2 - 2 r \beta + \beta^2 - \sigma^2)^2}{\sqrt{2\pi}\ \sigma^6 \sqrt{\sigma^2}}\, dr\right]$

$\mathtt{sJ23[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r^2 - 2 r \beta + \beta^2 - \sigma^2)}{\sqrt{2\pi}\ \sigma^3 \sqrt{\sigma^2}}\, dr\right]$

$\mathtt{sJ31[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r-\beta)}{\sqrt{2\pi}\ (\sigma^2)^{3/2}}\, dr\right]$

$\mathtt{sJ32[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}} (r^2 - 2 r \beta + \beta^2 - \sigma^2)}{\sqrt{2\pi}\ \sigma^3 \sqrt{\sigma^2}}\, dr\right]$

$\mathtt{sJ33[\beta\_,\ \sigma\_,\ C\_] := Assuming}\left[\sigma > 0\ \&\&\ r \in \mathtt{Integers},\ \int_{-\infty}^{\infty} \frac{e^{-\frac{(r-\beta)^2}{2\sigma^2}}}{C \sqrt{2\pi}\ \sqrt{\sigma^2}}\, dr\right]$

In[41]:= $\mathtt{sFI[\beta\_,\ \sigma\_,\ C\_] :=} \begin{pmatrix} \mathtt{sJ11[\beta,\ \sigma,\ C]} & \mathtt{sJ12[\beta,\ \sigma,\ C]} & \mathtt{sJ13[\beta,\ \sigma,\ C]} \\ \mathtt{sJ21[\beta,\ \sigma,\ C]} & \mathtt{sJ22[\beta,\ \sigma,\ C]} & \mathtt{sJ23[\beta,\ \sigma,\ C]} \\ \mathtt{sJ31[\beta,\ \sigma,\ C]} & \mathtt{sJ32[\beta,\ \sigma,\ C]} & \mathtt{sJ33[\beta,\ \sigma,\ C]} \end{pmatrix}$

In[42]:= $\mathtt{sCRLB[\beta\_,\ \sigma\_,\ C\_] := Inverse[sFI[\beta,\ \sigma,\ C]]}$

In[43]:= $\mathtt{MatrixForm[FullSimplify[sFI[\beta,\ \sigma,\ C]]]}$

Out[43]//MatrixForm=

$$\begin{pmatrix} \frac{C}{\sigma^2} & 0 & 0 \\ 0 & \frac{2C}{\sigma^2} & 0 \\ 0 & 0 & \frac{1}{C} \end{pmatrix}$$

In[44]:= $\mathtt{MatrixForm[FullSimplify[sCRLB[\beta,\ \sigma,\ C]]]}$

Out[44]//MatrixForm=

$$\begin{pmatrix} \frac{\sigma^2}{C} & 0 & 0 \\ 0 & \frac{\sigma^2}{2C} & 0 \\ 0 & 0 & C \end{pmatrix}$$

Appendix B: MATLAB Version of ML Algorithm

```
% Maximum Likelihood Estimate Using Optional Grid & Gradient Decent
Search
% function a = mliwb (Dv,Ivi,lIvi,N,aMax,Grd)
% Dv    - Data Vector
% Ivi   - Image Vector (Interpolated)
% lIvi  - Log of Interpolated Image Vector
% N     - Interpolation Points Used
% aMax  - Maximum Search Area = +/- aMax
% Grd   - Step Size for Grid Search 0 < Grd < length(Dv) to Perform

function a = mliwb (Dv,Ivi,lIvi,N,aMax,Grd)

Grd = floor(Grd*N)/N;                   % Ensure Grid Conforms to vecs
step = 1/N;                             % Interpolations Step Size

DvL  = length(Dv);                      % Data Vector Length
DvuL = (DvL-1)*N+1;                     % Upsampled Data Vector Length
IvuL = length(Ivi);                     % Interpolated Image Vector Len
IvL = (IvuL-1)/N+1;                     % Image Vector Length
if IvL < DvL            % Check for incorrect input vectors
    a = NaN;
    return;
elseif IvL <= 2*DvL     % Get Maximum Window Size
    sLim = (IvL/2-1)/2;
else
    sLim = (DvL-1)/2;
end;
wLim = sLim - (DvL - 2*sLim+1 - 1)/2;   % Get Minimum Window Size
if aMax > sLim          % check aMax is Less than Maximum Window
    aMax = sLim;
    Sl = 2*sLim - aMax;
elseif aMax < wLim      % Check aMax is Greater than Minimum Window
    Sl = sLim;
else
    Sl = 2*sLim - aMax;
end;
iSg= (IvL-1)/2;
iSs= (iSg-Sl+1-1)*N+1;      % start index for smaller image
iSe= (iSg+Sl+1-1)*N+1;      % end index for smaller image
dSg= (DvL-1)/2;
dSs= dSg-Sl+1;              % start index for smaller data
dSe= dSg+Sl+1;              % end index for smaller data
sSs= (iSg-aMax+1-1)*N+1;    % start index for minimum Shift (in window)
sSe= (iSg+aMax+1-1)*N+1;    % end index for maximum Shift (in window)

if Grd == 0 || Grd*N > sSe-sSs  % Determine if Grid Search is Required
    noGrid = true;
else
    noGrid = false;
end;

taL = zeros(1,IvuL);            % Allocate Memory (overkill)
```

```
if noGrid                          % Just Compute Endpoints (no Grid)
    for idx = [sSs sSe]                % Get LL Values for Endpoints
        taL(idx) = sum(Dv(dSs:dSe).*...
            lIvi((iSs:N:iSe)+(IvuL-1)/2-(idx-1))...
            -Ivi((iSs:N:iSe)+(IvuL-1)/2-(idx-1)),2);  % Shifts Image
    end;
    if taL(sSs) < taL(sSe)             % Right-Side is Bigger
        max = sSe;
        sdx = 2;
    else                               % Left-Side is Bigger
        max = sSs;
        sdx = 1;
    end;
    idx = sSs+floor((sSe-sSs)/2);      % Compute Next Search Point
else                               % Perform Grid Search
    init = sSs:Grd*N:sSe; max = sSs;
    if mod(sSe-sSs,Grd*N) ~= 0         % Add Last Point if Necessary
        init = [init sSe];
    end;
    for idx = init                     % Quick search for points on peak
        taL(idx) = sum(Dv(dSs:dSe).*...
            lIvi((iSs:N:iSe)+(IvuL-1)/2-(idx-1))...
            -Ivi((iSs:N:iSe)+(IvuL-1)/2-(idx-1)),2);
        if taL(idx) > taL(max)             % Replace Maximum if Necessary
            max = idx;
        end;
    end;
    if max == sSs                      % If Max is Left-Side
        taL(max+1) = sum(Dv(dSs:dSe).*...   % Get Next Value
              lIvi((iSs:N:iSe)+(IvuL-1)/2-(max+1-1))...
              -Ivi((iSs:N:iSe)+(IvuL-1)/2-(max+1-1)),2);  % Shifts Image
        if taL(max) < taL(max+1)           % Right-Side Bigger
            sSs = max+1;
            if max+Grd*N <= sSe                 % Decide if EndPoint
                sSe = max+Grd*N;
            else
                sSe = sSe;
            end;
            sdx = 1;
            idx = sSs+floor((sSe-sSs)/2);       % Compute Next Search Point
        else                               % Left-Side is Peak
            idx = -1;                           % Finish
        end;
    elseif max == sSe                  % If Max is Right-Side
        taL(max-1) = sum(Dv(dSs:dSe).*...   % Get Previous Value
              lIvi((iSs:N:iSe)+(IvuL-1)/2-(max-1-1))...
              -Ivi((iSs:N:iSe)+(IvuL-1)/2-(max-1-1)),2);  % Shifts Image
        if taL(max) < taL(max-1)           % Left-Side Bigger
            if mod(sSe-sSs,Grd*N) == 0          % Determine Previous Point
                sSs = max-Grd*N;
            else
                sSs = max-mod(sSe-sSs,Grd*N);
            end;
            sSe = max-1;
            sdx = 2;
            idx = sSs+floor((sSe-sSs)/2);       % Compute Next Search Point
        else                               % Right-Side is Peak
```

```
            idx = -1;                       % Finish
        end;
    else                            % Max is in Middle
        taL(max+1) = sum(Dv(dSs:dSe).*...   % Get Next Value
              lIvi((iSs:N:iSe)+(IvuL-1)/2-(max+1-1))...
              -Ivi((iSs:N:iSe)+(IvuL-1)/2-(max+1-1)),2);  % Shifts Image
        if taL(max) < taL(max+1)        % Right-Side is Bigger
            sSs = max+1;
            if max+Grd*N <= sSe             % Decide if EndPoint
                sSe = max+Grd*N;
            else
                sSe = sSe;
            end;
            sdx = 1;
            idx = sSs+floor((sSe-sSs)/2);   % Compute Next Search Point
        else                            % Left Side is Bigger
            sSs = max-Grd*N;
            sSe = max;
            sdx = 2;
            idx = sSs+floor((sSe-sSs)/2);   % Compute Next Search Point
        end;
    end;
end;

while idx > 0                   % Check for Complete Condition
    if idx <= sSs || idx >= sSe     % Check to See if we're Done
        idx = -1;                       % Finish
    else
        taL(idx)=sum(Dv(dSs:dSe).*...   % Compute Next LL Value
            lIvi((iSs:N:iSe)+(IvuL-1)/2-(idx-1))...
            -Ivi((iSs:N:iSe)+(IvuL-1)/2-(idx-1)),2);  % Shift Image

        if taL(idx) > taL(max)          % New Value Bigger than Old Max
            if idx+1 < sSe                  % Get Next Value
                taL(idx+1)=sum(Dv(dSs:dSe).*...
                    lIvi((iSs:N:iSe)+(IvuL-1)/2-((idx-1)+1))...
                    -Ivi((iSs:N:iSe)+(IvuL-1)/2-((idx-1)+1)),2);
            end;
            dir = taL(idx) < taL(idx+1);    % Store Sign of Slope
            if dir                              % Store Correct Index
                idx = idx+1;
            end;
            max = idx;                      % Store New Max
            if ~dir                         % Shutter Right (Peak Left)
%                 sSs = sSs;
                sSe = idx;
                sdx = 2;
            else                            % Shutter Left (Peak Right)
                sSs = idx;
%                 sSe = sSe;
                sdx = 1;
            end;
        else                            % Slope is Toward Current Max
%             max = max;
            if sdx == 1                     % Shutter Right (Peak Left)
%                 sSs = sSs;
                sSe = idx;
%                 sdx = 1;
```

```
            else                            % Shutter Left (Peak Right)
                sSs = idx;
%                 sSe = sSe;
%                 sdx = 2;
            end;
        end;
        idx = sSs+floor((sSe-sSs)/2);   % Compute Next Search Point
    end;
end;

a = (max-1-(IvuL-1)/2)/N;       % Convert to Shift Value (FloatingPoint)
```

C.1.Log-Likelihood Computation Module

```
LIBRARY ieee ;                                   -- Standard Includes
USE ieee.std_logic_1164.all ;
USE IEEE.std_logic_arith.all;
USE IEEE.std_logic_signed.all;

ENTITY LL IS
  GENERIC
  (
    N           : INTEGER := 128;          -- Number of Points Interpolated
--  DvL         : INTEGER := 32;           -- Length of Data
--  IvL         : INTEGER := 32;           -- Length of Image
--  DvuL        : INTEGER := 3969;         -- Length of Data (in UpSampled Terms)
    IvuL        : INTEGER := 3969;         -- Length of Image (in UpSampled Terms)
--  iSg         : INTEGER := 15.5;         -- Pixel Range of Half of Image
    iSs         : INTEGER := 1024;         -- Start of Window for Image (in UpSampled Terms)
--  iSe         : INTEGER := 2944;         -- End of Window for Image (in UpSampled Terms)
--  dSg         : INTEGER := 15.5;         -- Pixel Range of Half of Data
    dSs         : INTEGER := 8;            -- Start of Window for Data
    dSe         : INTEGER := 23;           -- End of Window for Data
--  sSs         : INTEGER := 1024;         -- First Possible Search (in UpSampled Terms)
--  sSe         : INTEGER := 2944;         -- Last Possible Search (in UpSampled Terms)
    BitDepth    : INTEGER :=  32           -- Number of Bits for Address and Data
  );
  PORT
  (
    Clock, Enable, Reset: IN  STD_LOGIC;                        -- Standard Signals
    Shift       : IN  INTEGER;                                  -- Shift Value
    Data        : IN  INTEGER;                                  -- Data Value
    Image       : IN  INTEGER;                                  -- Image Value
    lImage      : IN  INTEGER;                                  -- Log of Image Value
    DataAddr    : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);    -- Address for Data
    ImageAddr   : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);    -- Address for Image
    lImageAddr  : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);    -- Address for Log of Image (= ImageAddr)
    Result      : OUT INTEGER;                                  -- Log-Likelihood Result
    Valid       : OUT BOOLEAN                                   -- = TRUE When Finished
  );
END LL;

ARCHITECTURE extmem OF LL IS
BEGIN
  loglikelihood:                  -- Computes one Log-Likelihood
  PROCESS (Clock, Reset)
    VARIABLE  Idx : INTEGER := 0; -- Loop Variable
    VARIABLE  Ans : INTEGER := 0; -- Accumulates Answer
  BEGIN
    IF Reset = '1' THEN               -- Reset Calculator
      Valid   <=  FALSE;
      Result    <=  -1;                 -- Displays All 1's to Indicate Incorrect Answer
      Ans     :=  -1;                   -- "
      DataAddr  <=  (OTHERS => '1');    -- Displays All 1's to Indicate Incorrect Address
      ImageAddr <=  (OTHERS => '1');    -- "
      lImageAddr  <=  (OTHERS => '1');  -- "
      Idx     :=  0;                    -- Resets Counter for Use
    ELSIF Rising_Edge(Clock) THEN       -- Run For Loop on Clock
      IF Enable = '1' Then                -- Compute Only if Enabled
        IF (Idx + dSs) > dSe + 1 THEN       -- End State, Holds Current Result
          Valid   <=  TRUE;
          Result    <=  Ans;                  -- Outputs Answer
          Ans     :=  Ans;
          DataAddr  <=  (OTHERS => '1');      -- Displays All 1's to Indicate Incorrect Address
          ImageAddr <=  (OTHERS => '1');      -- "
          lImageAddr  <=  (OTHERS => '1');    -- "
          Idx     :=  Idx;                    -- Stays in "Hold" state
        ELSE                                -- Continue Computing Log-Likelihood
          IF Idx = 0 THEN                   -- If the Computation Just Started
            Valid   <=  FALSE;
            Result    <=  -1;                 -- Displays All 1's to Indicate Incorrect Answer
            Ans     :=  0;                    -- Resets Ans for Accumulation Use
            DataAddr  <=  CONV_STD_LOGIC_VECTOR(dSs + idx,BitDepth);
            ImageAddr <=  CONV_STD_LOGIC_VECTOR(iSs + idx*N + (IvuL-1)/2 - (Shift - 1),BitDepth);
            lImageAddr  <=  CONV_STD_LOGIC_VECTOR(iSs + idx*N + (IvuL-1)/2 - (Shift - 1),BitDepth);
            Idx     :=  Idx + 1;              -- Increment for Next Clock Cycle
          ELSE                              -- If in Middle of Computation
            Ans :=  Ans + Data * lImage - Image;    -- ***** Compute and Add Current Value to Ans *****
            IF (Idx + dSs) > dSe THEN         -- If the Current Value is the End
```

```
              Valid   <=  TRUE;
              Result    <=  Ans;                -- Early Output of Answer (Saves 1 Clock Cycle)
              DataAddr  <=  (OTHERS => '1');    -- Displays All 1's to Indicate Incorrect Address
              ImageAddr <=  (OTHERS => '1');    -- "
              lImageAddr  <=  (OTHERS => '1');  -- "
            ELSE                              -- Continues Computation
              Valid   <=  FALSE;
              Result    <=  -1;                 -- Displays All 1's to Indicate Incorrect Answer
              DataAddr  <=  CONV_STD_LOGIC_VECTOR(dSs + idx,BitDepth);
              ImageAddr <=  CONV_STD_LOGIC_VECTOR(iSs + idx*N + (IvuL-1)/2 - (Shift - 1),BitDepth);
              lImageAddr  <=  CONV_STD_LOGIC_VECTOR(iSs + idx*N + (IvuL-1)/2 - (Shift - 1),BitDepth);
            END IF;
            Idx     :=  Idx + 1;              -- Increment for Next Clock Cycle
          END IF;
        END IF;
      ELSE                                -- Reset Calculator for Next Computation
        Valid   <=  FALSE;
        Result    <=  -1;                   -- Displays All 1's to Indicate Incorrect Answer
        Ans     :=  -1;                     -- "
        DataAddr  <=  (OTHERS => '1');      -- Displays All 1's to Indicate Incorrect Address
        ImageAddr <=  (OTHERS => '1');      -- "
        lImageAddr  <=  (OTHERS => '1');    -- "
        Idx     :=  0;                      -- Resets Counter for ReUse
      END IF;
    END IF;
  END PROCESS loglikelihood;
END extmem;
```

C.2.Main Search Module

```
LIBRARY ieee ;                                    -- Standard Includes
USE ieee.std_logic_1164.all ;
USE IEEE.std_logic_arith.all;
USE IEEE.std_logic_signed.all;

ENTITY MLIW IS
  GENERIC
  (
    N           : INTEGER := 128;                 -- Number of Points Interpolated
--  DvL         : INTEGER := 32;                  -- Length of Data
--  IvL         : INTEGER := 32;                  -- Length of Image
--  DvuL        : INTEGER := 3969;                -- Length of Data (in UpSampled Terms)
    IvuL        : INTEGER := 3969;                -- Length of Image (in UpSampled Terms)
--  iSg         : INTEGER := 15.5;                -- Pixel Range of Half of Image
    iSs         : INTEGER := 1024;                -- Start of Window for Image (in UpSampled Terms)
--  iSe         : INTEGER := 2944;                -- End of Window for Image (in UpSampled Terms)
--  dSg         : INTEGER := 15.5;                -- Pixel Range of Half of Data
    dSs         : INTEGER := 8;                   -- Start of Window for Data
    dSe         : INTEGER := 23;                  -- End of Window for Data
    sSs         : INTEGER := 1024;                -- First Possible Search (in UpSampled Terms)
    sSe         : INTEGER := 2944;                -- Last Possible Search (in UpSampled Terms)
    Step        : INTEGER := 960;                 -- Step Size for Grid Search
    Grid        : BOOLEAN := FALSE;               -- Determines if Grid Search Happens
    BitDepth    : INTEGER :=  32                  -- Number of Bits for Address and Data
  );
  PORT
  (
    Clock, Enable, Reset: IN  STD_LOGIC;                          -- Standard Signals
    Data        : IN  STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);  -- Data Value
    Image       : IN  STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);  -- Image Value (Multiplied by N or more)
    lImage      : IN  STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);  -- Log of Image Value (Multiplied by N+)
    DataAddr    : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);  -- Address for Data
    ImageAddr   : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);  -- Address for Image
    lImageAddr  : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);  -- Address for Log of Image (= ImageAddr)
    Result      : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);  -- Result: Shift = (Result-1-(IvuL-1)/2)/N
    Valid       : OUT STD_LOGIC                               -- = TRUE When Finished
  );
```

```
END MLIW;

ARCHITECTURE extmem OF MLIW IS
  COMPONENT LL
    GENERIC
    (
      N            : INTEGER := 128;
  --  DvL          : INTEGER := 32;
  --  IvL          : INTEGER := 32;
  --  DvuL         : INTEGER := 3969;
      IvuL         : INTEGER := 3969;
  --  iSg          : INTEGER := 15.5;
      iSs          : INTEGER := 1024;
  --  iSe          : INTEGER := 2944;
  --  dSg          : INTEGER := 15.5;
      dSs          : INTEGER := 8;
      dSe          : INTEGER := 23;
  --  sSs          : INTEGER := 1024;
  --  sSe          : INTEGER := 2944;
      BitDepth     : INTEGER :=  32
    );
    PORT
    (
      Clock, Enable, Reset: IN  STD_LOGIC;
      Shift        : IN  INTEGER;
      Data         : IN  INTEGER;
      Image        : IN  INTEGER;
      lImage       : IN  INTEGER;
      DataAddr     : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);
      ImageAddr    : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);
      lImageAddr   : OUT STD_LOGIC_VECTOR(BitDepth-1 DOWNTO 0);
      Result       : OUT INTEGER;
      Valid        : OUT BOOLEAN
    );
  END COMPONENT;

  TYPE    STATE_TYPE    IS  (Start, PreCompute, Compute, Hold); -- Extensible State Type
  TYPE    INT_ARRAY_TYPE  IS ARRAY (3 DOWNTO 0) OF INTEGER;     -- INTEGER Array Type
  SIGNAL    State   : STATE_TYPE;                               -- State Machine Variable
  SIGNAL    Answer  : INTEGER;                                  -- Placeholder Result (for Moore Machine)
  SIGNAL    Complete: BOOLEAN;                                  -- Internal State Transition Variable
  SIGNAL    lEnable : STD_LOGIC;                                -- Enable for Log-Likelihood Calculator
  SIGNAL    lReset  : STD_LOGIC;                                -- Reset for "
  SIGNAL    lIdx    : INTEGER;                                  -- Shift for "
  SIGNAL    lResult : INTEGER;                                  -- Result from "
  SIGNAL    lValid  : BOOLEAN;                                  -- Validity from "

BEGIN
  LL1 : LL
  GENERIC MAP
  (
    N           =>  N,
--  DvL         =>  DvL,
--  IvL         =>  IvL,
--  DvuL        =>  DvuL,
    IvuL        =>  IvuL,
--  iSg         =>  iSg,
    iSs         =>  iSs,
--  iSe         =>  iSe,
--  dSg         =>  dSg,
    dSs         =>  dSs,
    dSe         =>  dSe,
--  sSs         =>  sSs,
--  sSe         =>  sSe,
    BitDepth    =>  BitDepth
  )
  PORT MAP
  (
    Clock => Clock, Enable => lEnable, Reset => lReset,
    Shift   => lIdx,
    Data    => CONV_INTEGER(Data),
    Image   => CONV_INTEGER(Image),
    lImage    => CONV_INTEGER(lImage),
    DataAddr  => DataAddr,
    ImageAddr => ImageAddr,
    lImageAddr  => lImageAddr,
    Valid   => lValid,
    Result    => lResult
  );
```

```
statemachine:                       -- Moore State Machine to Control Calcuation
PROCESS (Clock, Reset)
BEGIN
  IF Reset = '1' THEN                 -- Reset State: Mealy Includes Hold Here
    State       <=  Start;              -- Moves Immediately to Start State
  ELSIF Rising_Edge(Clock) THEN
    CASE State IS
      WHEN Start    =>                -- Start State: Prepares for Calculation
        IF Enable = '1' THEN            -- Waits until Enable to Move to PreCompute
          State <=  PreCompute;
        END IF;
      WHEN PreCompute =>              -- PreCompute: Computes EndPoints or Grid Search
        IF Complete THEN                -- Waits until Complete to Move to Compute
          State <=  Compute;
        END IF;
      WHEN Compute  =>                -- Compute: Performs Gradient-Decent Search
        IF Complete THEN                -- Waits until Complete to Move to Hold
          State <=  Hold;
        END IF;
      WHEN Hold   =>                  -- Hold: Outputs Answer, Prepares for Calculation
        IF Enable = '1' THEN            -- Waits until Enable to Move to PreCompute
          State <=  PreCompute;
        END IF;
      WHEN OTHERS   =>                -- Should NEVER Happen
        State   <=  Start;
    END CASE;
  END IF;
END PROCESS statemachine;

validout:                             -- Moore Output of Valid
WITH State SELECT
Valid <=  '1' WHEN  Hold,               -- Only in Hold State
      '0' WHEN  OTHERS;
resultout:                            -- Moore Output of Result
WITH State SELECT
Result  <=  CONV_STD_LOGIC_VECTOR(Answer,BitDepth)  WHEN  Hold, -- Only in Hold State
      (OTHERS => '1')            WHEN  OTHERS;                 -- Displays 1's for Incorrect Answer

binarysearch:                 -- Performs Grid and Binary Searches
PROCESS (Clock, Reset, lValid)
```

```
  VARIABLE  Idx   : INTEGER := 0;               -- Index for Next Shift
  VARIABLE  NeedR : BOOLEAN := FALSE;           -- Need Right Value in Grid Search
  VARIABLE  NeedS : BOOLEAN := FALSE;           -- Need Slope
  VARIABLE  LLVals  : INT_ARRAY_TYPE;           -- LL Values
  VARIABLE  LLIdxs  : INT_ARRAY_TYPE;           -- LL Indexes
  VARIABLE  LLMax : INTEGER;                    -- Current Max LL Index
BEGIN
  IF Reset = '1' THEN           -- Resets All Signals
    Idx     :=  0;
    NeedR   :=  FALSE;
    NeedS   :=  FALSE;
    LLVals    :=  (OTHERS => 0);
    LLIdxs    :=  (OTHERS => 0);
    LLMax   :=  0;
    Complete  <=  FALSE;
    Answer    <=  0;
    lEnable   <=  '0';
    lReset    <=  '1';
    lIdx    <=  Idx;
  ELSIF Rising_Edge(Clock) THEN -- Performs Operations
    CASE State IS
      WHEN Start  =>              -- Clear Everything to Begin
        Idx     :=  sSs;
        NeedR   :=  FALSE;
        NeedS   :=  FALSE;
        LLVals    :=  (OTHERS => 0);
        LLIdxs    :=  (OTHERS => 0);
        LLMax   :=  0;
        Complete  <=  FALSE;
        Answer    <=  0;
        lEnable   <=  '0';
        lReset    <=  '0';
        lIdx    <=  Idx;
      WHEN PreCompute =>          -- Computes EndPoints or Grid Search
        IF NOT Complete THEN        -- Wait for Loop to Complete
          IF NOT lValid THEN          -- Wait for LL Calc to Complete
            Idx     :=  Idx;
            NeedR   :=  NeedR;
            NeedS   :=  NeedS;
            LLVals    :=  LLVals;
  LLIdxs    :=  LLIdxs;
  LLMax   :=  LLMax;
  Complete  <=  FALSE;
  lEnable   <=  '1';
ELSIF lEnable = '1' THEN  -- End of 1 LL computation
  IF NOT NeedS THEN         -- Don't Need the Slope
    IF Idx = sSs THEN         -- Just store if it's first
      IF NOT Grid THEN          -- Select Next Point
        Idx   := sSe;
      ELSE
        Idx   :=  Idx + Step;
      END IF;
      NeedR   :=  TRUE;
      NeedS   :=  FALSE;
      LLVals    :=  (OTHERS => lResult);
      LLIdxs    :=  (OTHERS => sSs);
      LLMax   :=  0;
      Complete  <=  FALSE;
      lEnable   <=  '0';        -- Clear the LL Calculator
    ELSE                      -- Not First Point
      IF LLVals(LLMax) < lResult THEN -- Result is Bigger
        LLVals(3 DOWNTO 0)      :=  (0 => LLVals(3),  1 => lResult,
                         2 => LLVals(2),  3 => lResult);
        LLIdxs(3 DOWNTO 0)      :=  (0 => LLIdxs(3),  1 => Idx,
                         2 => LLIdxs(2),  3 => Idx);
        LLMax           :=  1;
        IF Idx + Step <= sSe THEN
          NeedR         :=  TRUE;
        ELSE
          NeedR         :=  FALSE;
        END IF;
      ELSE                      -- Result is Smaller
        IF NeedR THEN
          IF LLMax = 0 THEN
            LLVals(3 DOWNTO 0)  :=  (0 => LLVals(0),  1 => lResult,
                         2 => LLVals(2),  3 => lResult);
            LLIdxs(3 DOWNTO 0)  :=  (0 => LLIdxs(0),  1 => Idx,
                         2 => LLIdxs(2),  3 => Idx);
          ELSE
            LLVals(3 DOWNTO 0)  :=  (0 => LLVals(0),  1 => LLVals(1),
                       2 => lResult,    3 => lResult);
          LLIdxs(3 DOWNTO 0)  :=  (0 => LLIdxs(0),  1 => LLIdxs(1),
                       2 => Idx,      3 => Idx);
        END IF;
      ELSE
        LLVals(3 DOWNTO 0)    :=  (0 => LLVals(0),  1 => LLVals(1),
                       2 => LLVals(2),  3 => lResult);
        LLIdxs(3 DOWNTO 0)    :=  (0 => LLIdxs(0),  1 => LLIdxs(1),
                       2 => LLIdxs(2),  3 => Idx);
      END IF;
      NeedR           :=  FALSE;
      LLMax           :=  LLMax;
    END IF;
    IF NOT Grid THEN        -- Complete the EndPoints
      NeedS           :=  FALSE;
      Complete          <=  TRUE;
    ELSE                    -- Continue with Grid Search
      IF Idx >= sSe THEN      -- Setup for Slope Calc
        IF LLIdxs(LLMax) = sSe THEN
          Idx         :=  LLIdxs(LLMax)-1;
        ELSE
          Idx         :=  LLIdxs(LLMax)+1;
        END IF;
        NeedS         :=  TRUE;
      ELSE                    -- Compute Next Point
        Idx           :=  Idx + Step;
        NeedS         :=  FALSE;
      END IF;
      Complete          <=  FALSE;
    END IF;
    lEnable             <=  '0';
  END IF;
ELSE                      -- Decide Window on Slope
  NeedR         :=  FALSE;
  NeedS         :=  FALSE;
  IF LLIdxs(LLMax) = sSs THEN -- Start of Window
    IF LLVals(LLMax) < lResult THEN
      Idx         :=  Idx;
      LLVals(1 DOWNTO 0)  :=  (0 => lResult,      1 => LLVals(1));
      LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(LLMax)+1,  1 => LLIdxs(1));
```

```
          LLMax        :=  0;
        ELSE
          Idx          :=  -1;
          LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => LLVals(1));
          LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => LLIdxs(1));
          LLMax        :=  LLMax;
        END IF;
      ELSIF LLIdxs(LLMax) = sSe THEN  -- End of Window
        IF LLVals(LLMax) < lResult THEN
          Idx          :=  Idx;
          LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => lResult);
          LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => LLIdxs(LLMax)-1);
          LLMax        :=  1;
        ELSE
          Idx          :=  -1;
          LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => LLVals(1));
          LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => LLIdxs(1));
          LLMax        :=  LLMax;
        END IF;
      ELSE                      -- Middle of Window
        Idx            :=  Idx;
        IF LLVals(LLMax) < lResult THEN
          LLVals(1 DOWNTO 0)  :=  (0 => lResult,      1 => LLVals(2));
          LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(LLMax)+1,  1 => LLIdxs(2));
          LLMax        :=  0;
        ELSE
          LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => LLVals(1));
          LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => LLIdxs(1));
          LLMax        :=  1;
        END IF;
      END IF;
      LLVals(3 DOWNTO 2)    :=  LLVals(3 DOWNTO 2);
      LLIdxs(3 DOWNTO 2)    :=  LLIdxs(3 DOWNTO 2);
      Complete  <=  TRUE;
      lEnable   <=  '0';
    END IF;
  ELSE                        -- Wait for LL Calc to Clear
    Idx     :=  Idx;
    NeedR   :=  NeedR;
    NeedS   :=  NeedS;
```

```
      LLVals    :=  LLVals;
      LLIdxs    :=  LLIdxs;
      LLMax   :=  LLMax;
      Complete  <=  FALSE;
      lEnable   <=  lEnable;
    END IF;
  ELSE                        -- Leaving State - Setup for Compute
    IF  Idx = -1 THEN
      Idx   :=  Idx;
    ELSE
      Idx   :=  (LLIdxs(0)+LLIdxs(1))/2;
    END IF;
    NeedR   :=  FALSE;
    NeedS   :=  FALSE;
    LLVals    :=  LLVals;
    LLIdxs    :=  LLIdxs;
    LLMax   :=  LLMax;
    Complete  <=  FALSE;
    lEnable   <=  '0';
  END IF;
  Answer    <=  0;
  lReset    <=  '0';
  lIdx    <=  Idx;

WHEN Compute  =>            -- Performs Full Search
  IF NOT Complete THEN        -- Wait for Loop to Complete
    IF NOT lValid THEN          -- Wait for LL Calc to Complete
      Idx     :=  Idx;
      NeedS   :=  NeedS;
      LLVals    :=  LLVals;
      LLIdxs    :=  LLIdxs;
      LLMax   :=  LLMax;
      IF Idx = -1 THEN
        Complete  <=  TRUE;
        lEnable   <=  '0';
      ELSE
        Complete  <=  FALSE;
        lEnable   <=  '1';
      END IF;
    ELSIF lEnable = '1' THEN    -- End of 1 LL computation
```

```
IF NOT NeedS THEN           -- Do Pre-Slope Computations
  IF LLVals(LLMax) < lResult THEN -- Result is Bigger
    LLVals(2) :=  lResult;
    LLIdxs(2) :=  Idx;
    IF Idx + 1 < LLIdxs(1) THEN   -- Setup for Slope Calc
      NeedS       :=  TRUE;
      LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => LLVals(1));
      LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => LLIdxs(1));
      LLMax       :=  LLMax;
      Idx         :=  Idx + 1;
    ELSE                          -- To Left, Shutter Right
      NeedS       :=  FALSE;
      LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => lResult);
      LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => Idx);
      LLMax       :=  1;
      Idx         :=  (LLIdxs(0)+Idx)/2;
    END IF;
  ELSE                          -- Result is Smaller
    NeedS         :=  FALSE;
    LLVals(2) :=  LLVals(2);
    LLIdxs(2) :=  LLIdxs(2);
    IF LLMax = 0 THEN             -- To Left, Shutter Right
      LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => lResult);
      LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => Idx);
      Idx         :=  (LLIdxs(0)+Idx)/2;
    ELSE                          -- To Right, Shutter Left
      LLVals(1 DOWNTO 0)  :=  (0 => lResult,  1 => LLVals(1));
      LLIdxs(1 DOWNTO 0)  :=  (0 => Idx,    1 => LLIdxs(1));
      Idx         :=  (Idx+LLIdxs(1))/2;
    END IF;
    LLMax         :=  LLMax;
  END IF;
  lEnable   <=  '0';            -- Clear LL Calculator
ELSE                          -- Do Post-Slope Computations
  NeedS   :=  FALSE;
  IF LLVals(2) < lResult THEN   -- To Right, Shutter Left
    LLVals(1 DOWNTO 0)  :=  (0 => lResult,  1 => LLVals(1));
    LLIdxs(1 DOWNTO 0)  :=  (0 => Idx,    1 => LLIdxs(1));
    LLMax       :=  0;
    Idx         :=  (Idx+LLIdxs(1))/2;
```

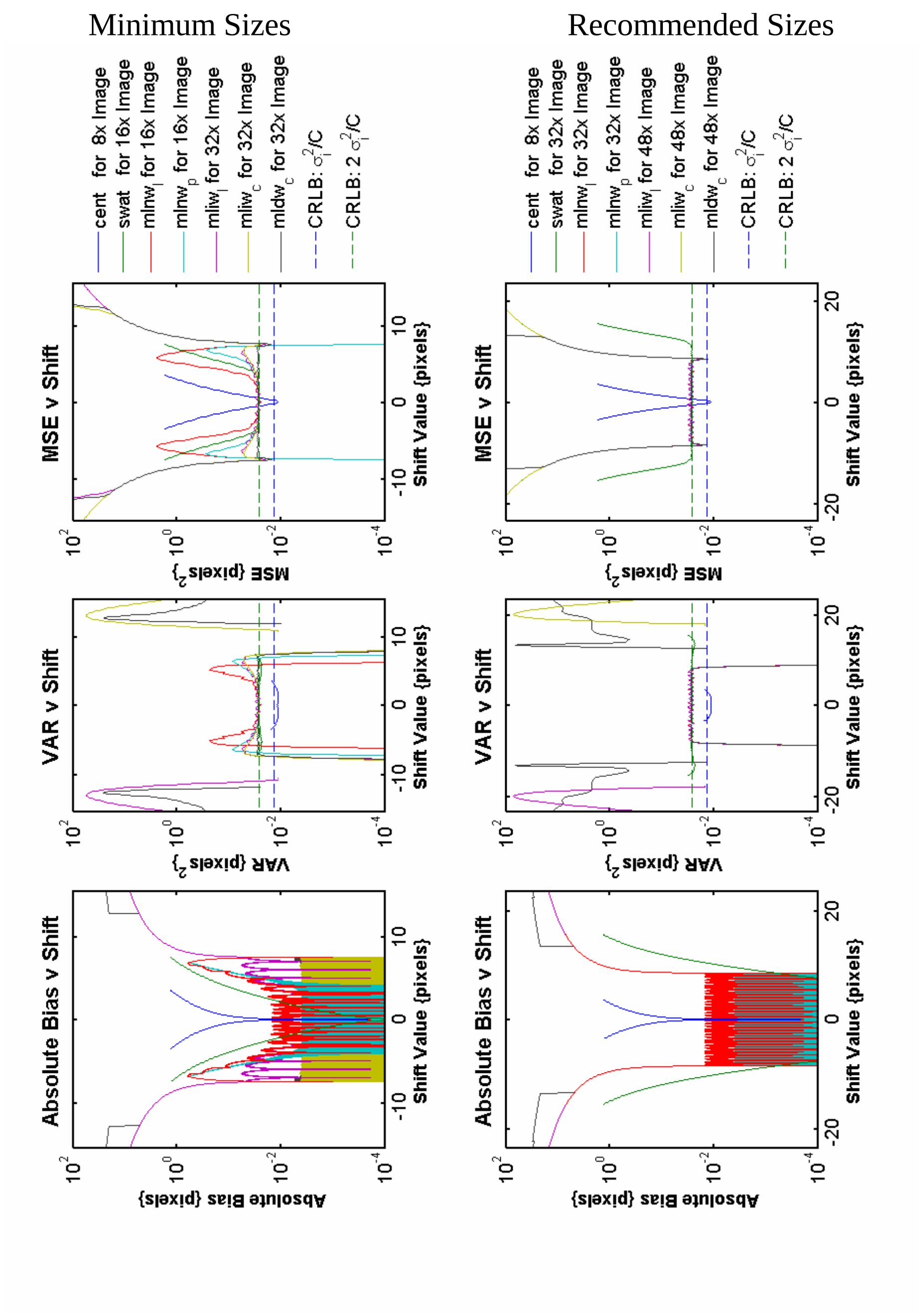
```
      ELSE                          -- To Left, Shutter Right
        LLVals(1 DOWNTO 0)  :=  (0 => LLVals(0), 1 => LLVals(2));
        LLIdxs(1 DOWNTO 0)  :=  (0 => LLIdxs(0), 1 => LLIdxs(2));
        LLMax       :=  1;
        Idx         :=  (LLIdxs(0)+LLIdxs(2))/2;
      END IF;
      LLVals(2) :=  LLVals(2);
      LLIdxs(2) :=  LLIdxs(2);
    END IF;
    LLVals(3) :=  LLVals(3);
    LLIdxs(3) :=  LLIdxs(3);
    IF Idx > LLIdxs(0) AND Idx < LLIdxs(1) THEN -- Continue Search
      Complete  <=  FALSE;
    ELSE                          -- End of Search
      Complete  <=  TRUE;
    END IF;
      lEnable   <=  '0';
  ELSE                          -- Wait for LL Calculator to Clear
    Idx     :=  Idx;
    NeedS   :=  NeedS;
    LLVals    :=  LLVals;
    LLIdxs    :=  LLIdxs;
    LLMax   :=  LLMax;
    Complete  <=  FALSE;
    lEnable   <=  lEnable;
  END IF;
ELSE                          -- Leaving State - Setup for Hold
  Idx     :=  Idx;
  NeedS   :=  FALSE;
  LLVals    :=  LLVals;
  LLIdxs    :=  LLIdxs;
  LLMax   :=  LLMax;
  Complete  <=  FALSE;
  lEnable   <=  '0';
END IF;
NeedR   :=  FALSE;
Answer    <=  0;
lReset    <=  '0';
lIdx    <=  Idx;
```

```
        WHEN Hold =>                -- Hold Output Answer
          Idx      :=  0;
          NeedR    :=  FALSE;
          NeedS    :=  FALSE;
          LLVals     :=  LLVals;
          LLIdxs     :=  LLIdxs;
          LLMax    :=  LLMax;
          Complete  <=  FALSE;
          Answer    <=  LLIdxs(LLMax);
          lEnable   <=  '0';
          lReset    <=  '0';
          lIdx    <=  Idx;

        WHEN OTHERS =>          -- Should Never Happen
          Idx      :=  0;
          NeedR    :=  FALSE;
          NeedS    :=  FALSE;
          LLVals     :=  LLVals;
          LLIdxs     :=  LLIdxs;
          LLMax    :=  LLMax;
          Complete  <=  Complete;
          Answer    <=  Answer;
          lEnable   <=  '0';
          lReset    <=  '0';
          lIdx    <=  Idx;
      END CASE;
    END IF;
  END PROCESS binarysearch;

END extmem;
```

Appendix D: Complete Parameterization of Bias and Noise Statistics

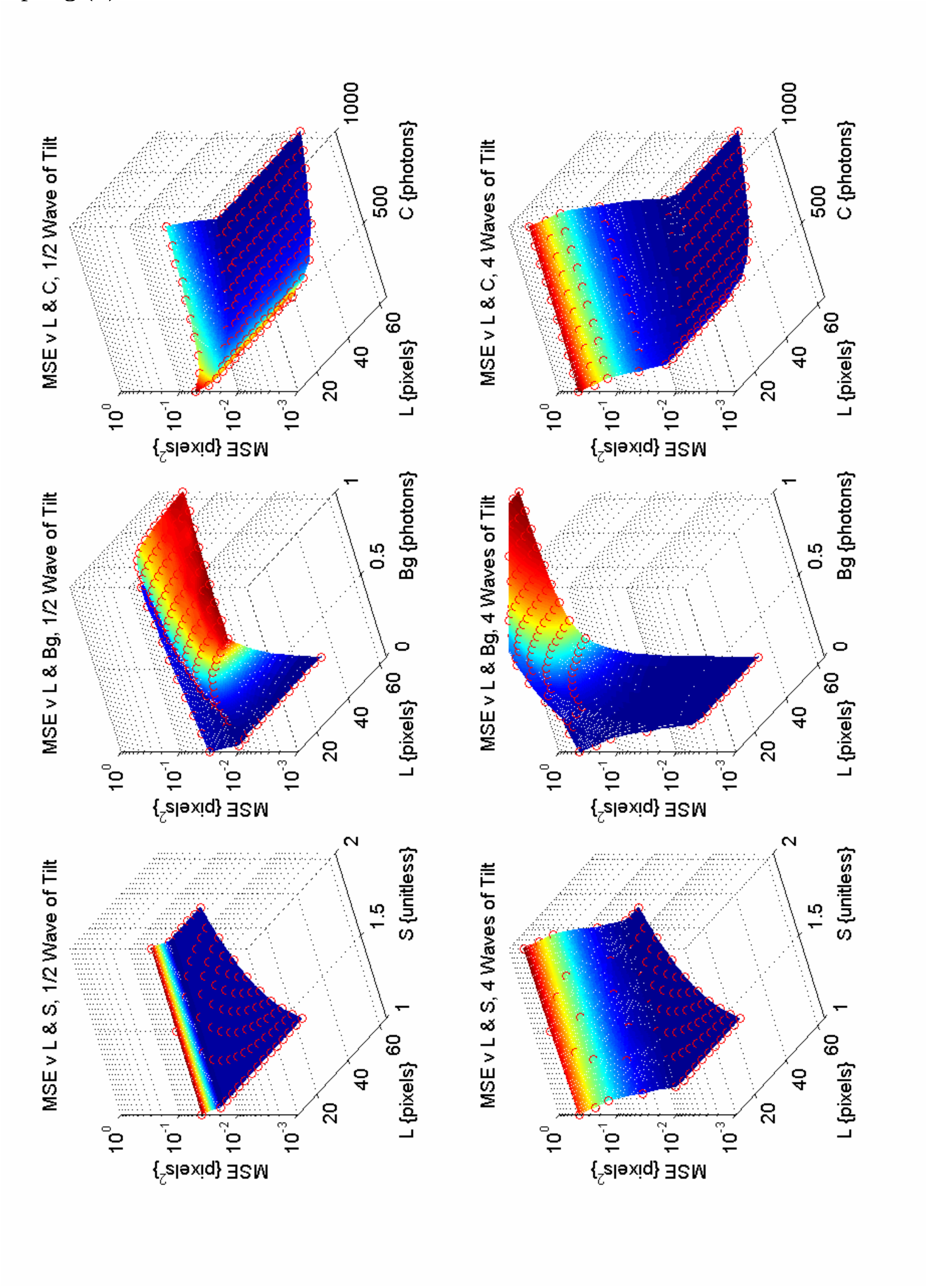
D.1. LGS Image Size Comparison

Parameters: C = 300, $\sigma_i = 2$, Bg = 0, 1xNyquist



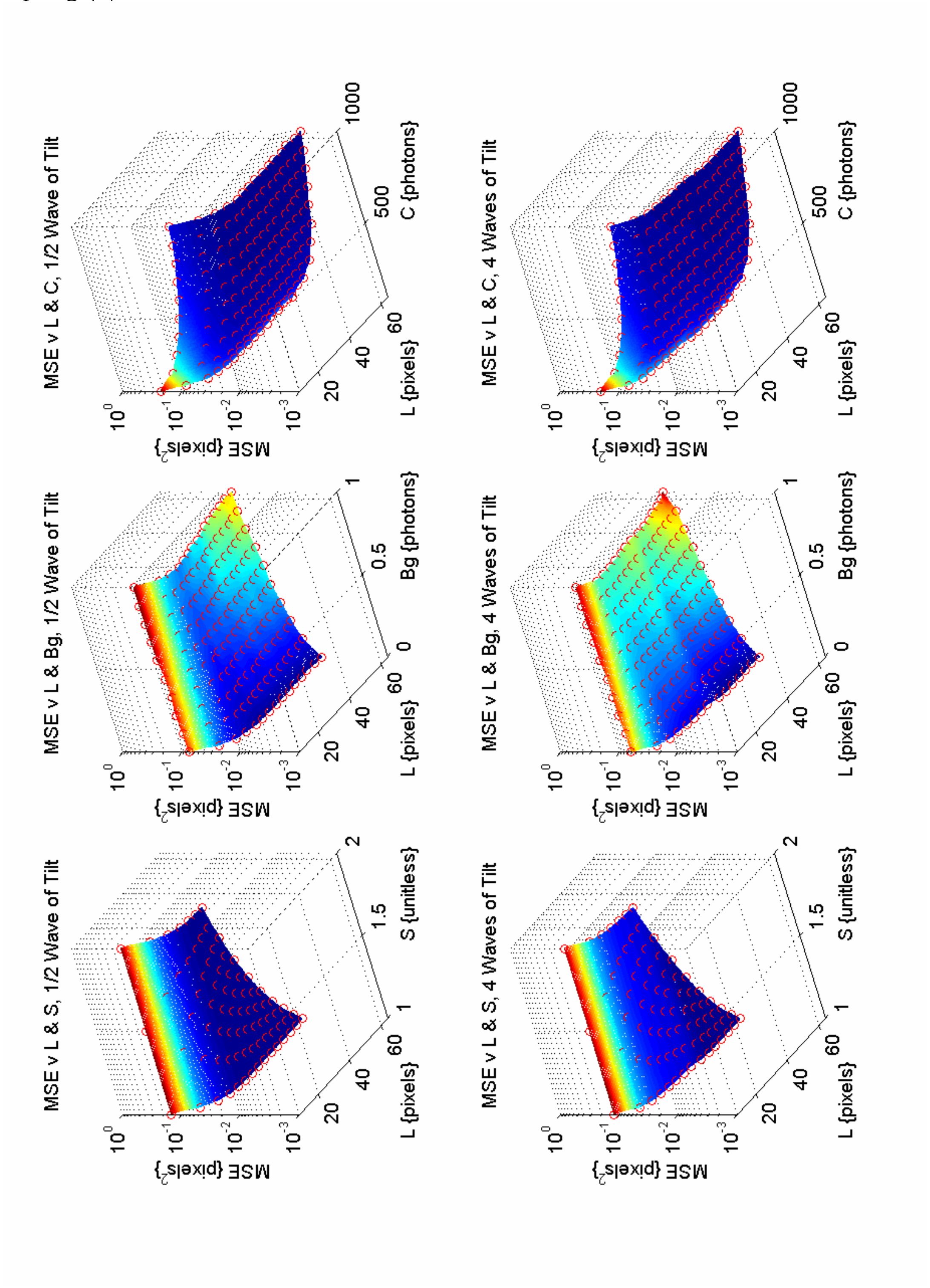
D.2. LGS swat Characterization.

Parameters: Image Size (L), Intensity (C) = 300, σ_i = 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.



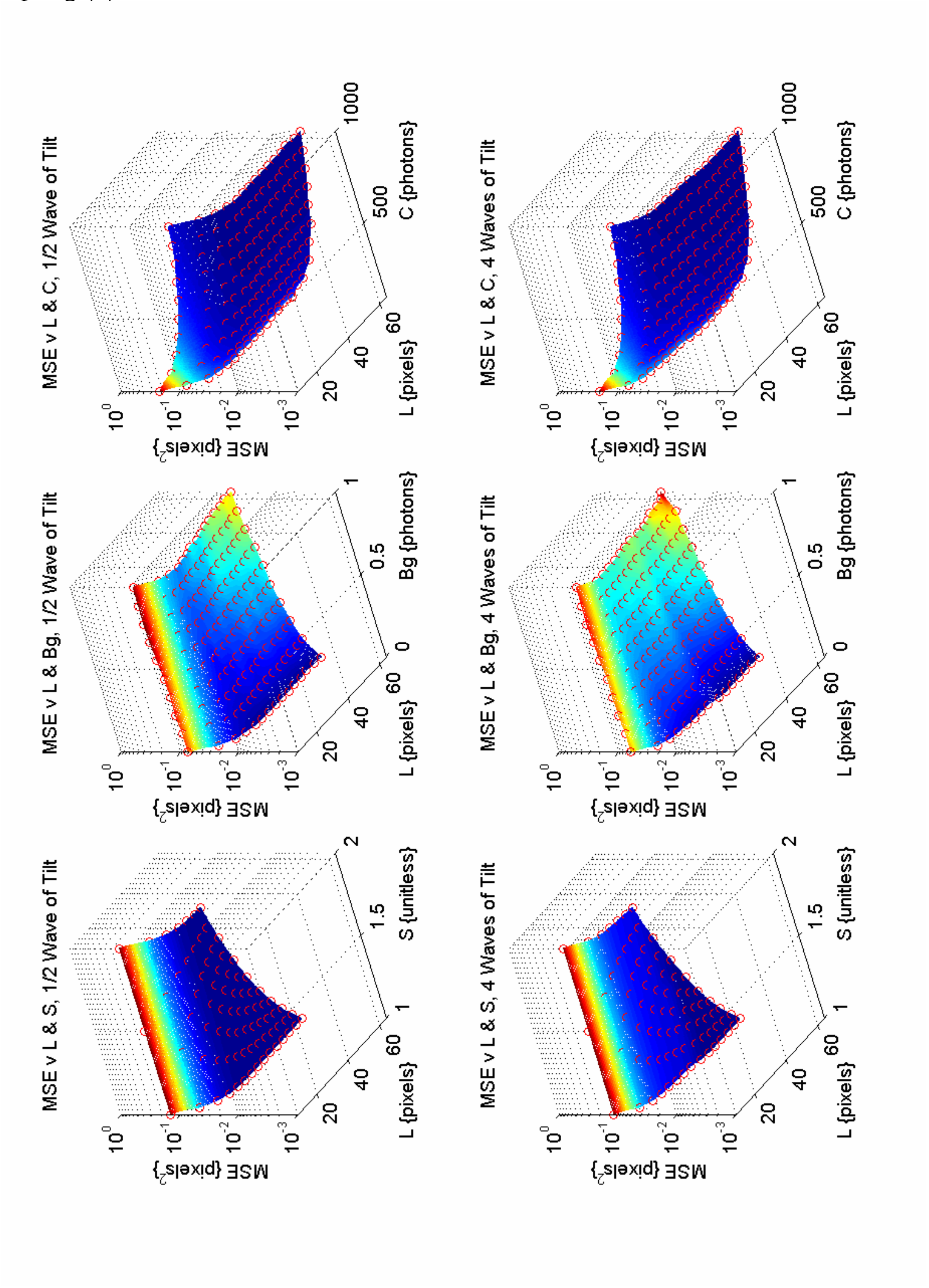
D.3. LGS mliw$_l$ Characterization.

Parameters: Image Size (L), Intensity (C) = 300, σ_i = 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.



D.4. LGS mliw$_c$ Characterization.

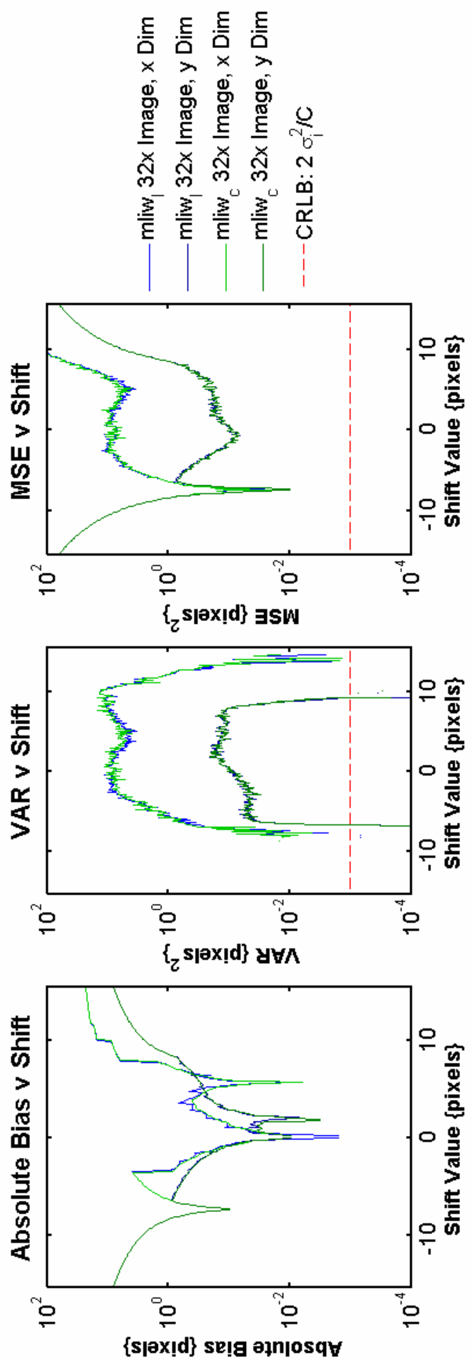
Parameters: Image Size (L), Intensity (C) = 300, σ_i = 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.



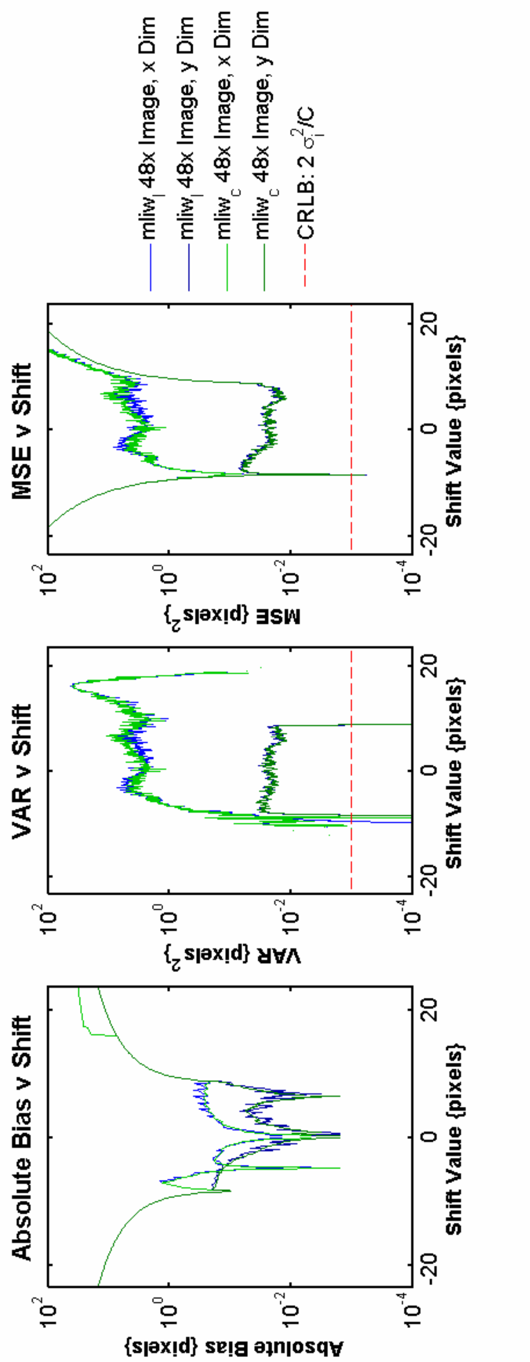
D.5. Tracking Hubble Image Size Comparison.

Parameters: C=8000, $\sigma_i \approx 2$, Bg=0, 1xNyquist

Minimum Sizes

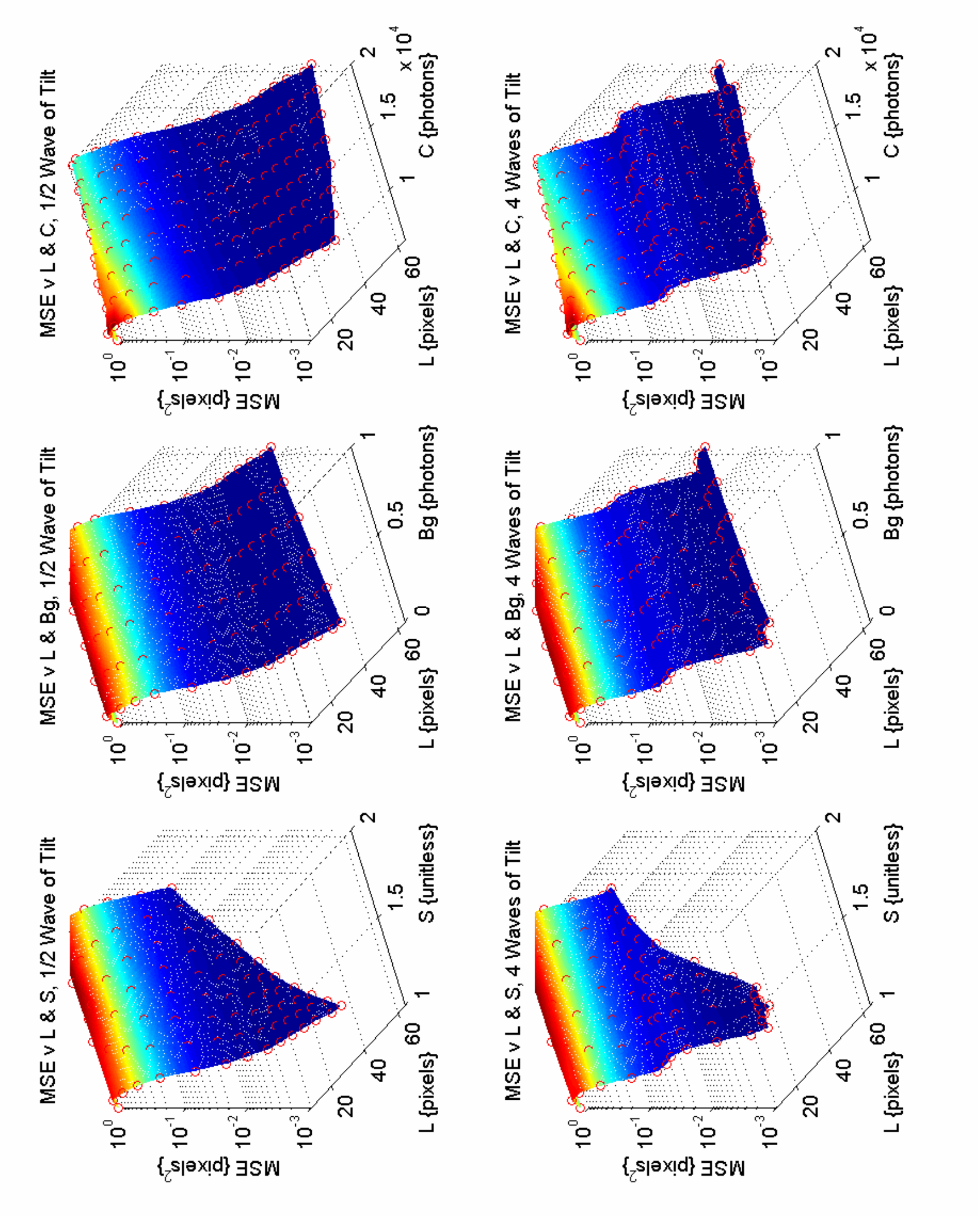


Recommended Sizes



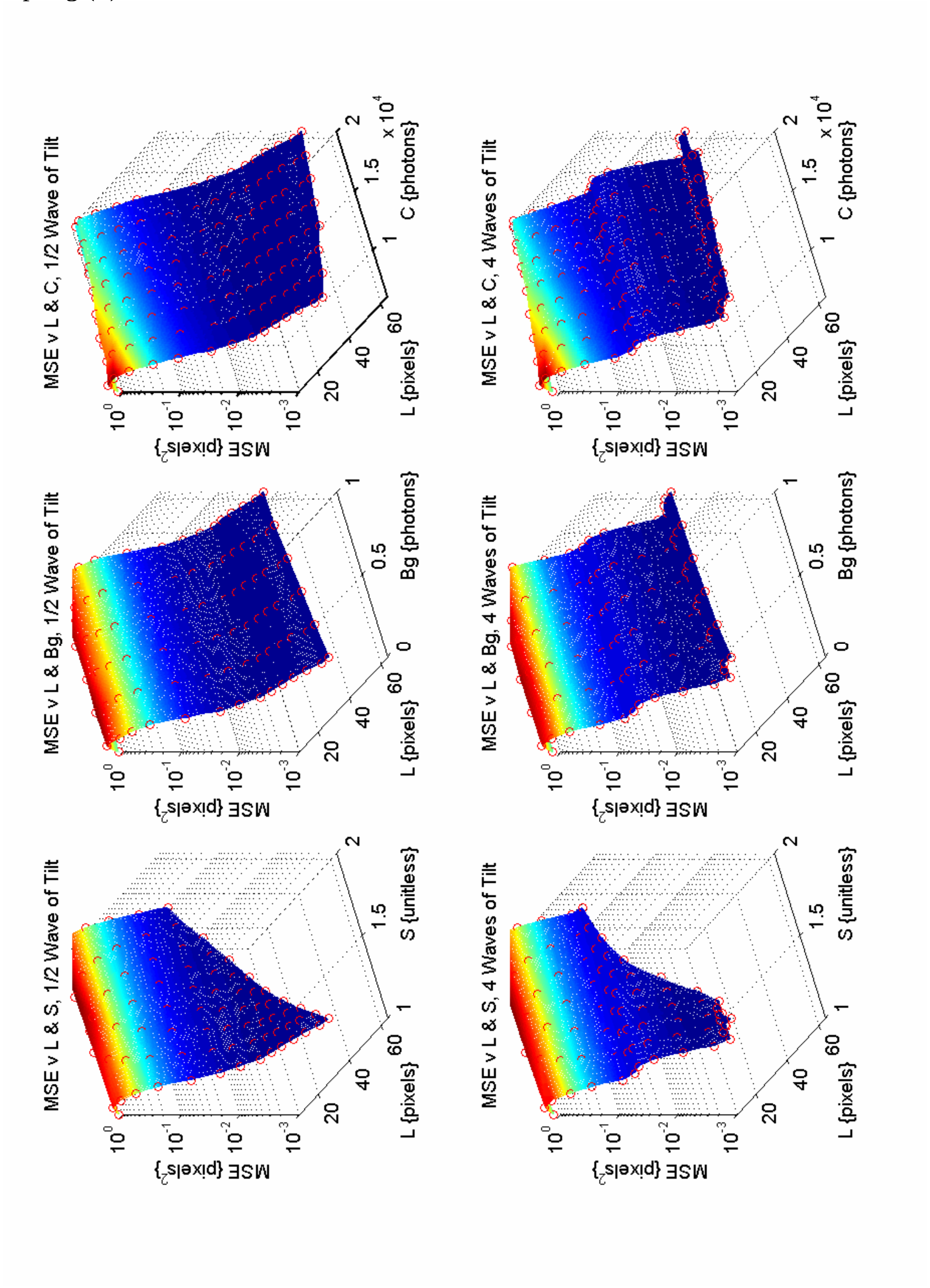
D.6. Tracking Hubble mliw$_l$ Characterization (y dim).

Parameters: Image Size (L), Intensity (C) = 8000, σ_i= 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.



D.7. Tracking Hubble mliw$_c$ Characterization (y dim).

Parameters: Image Size (L), Intensity (C) = 300, σ_i = 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.

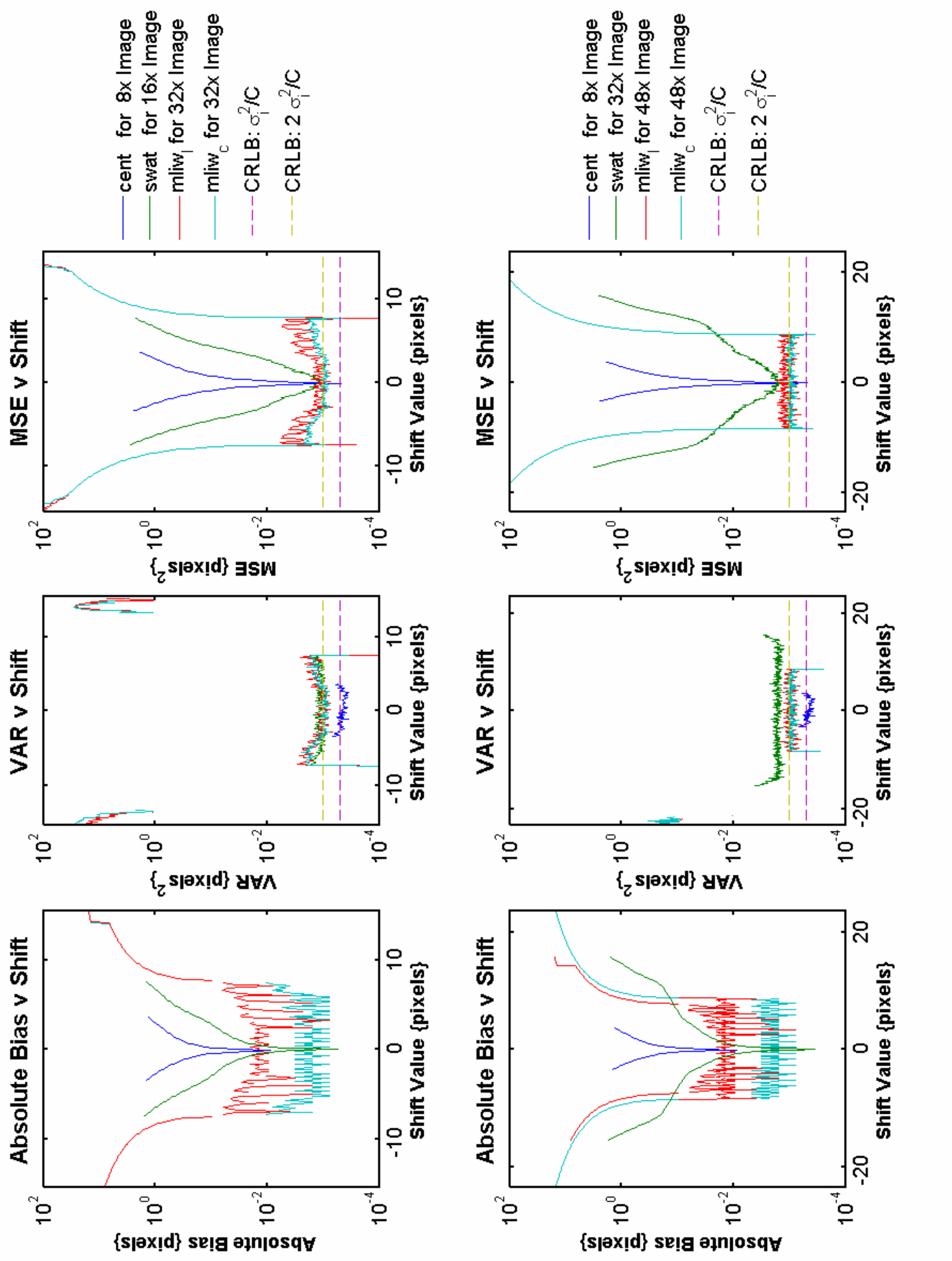


D.8. WFS Hubble Image Size Comparison.

Parameters: C=300, $\sigma_i \approx 2$, Bg=0, 1xNyquist

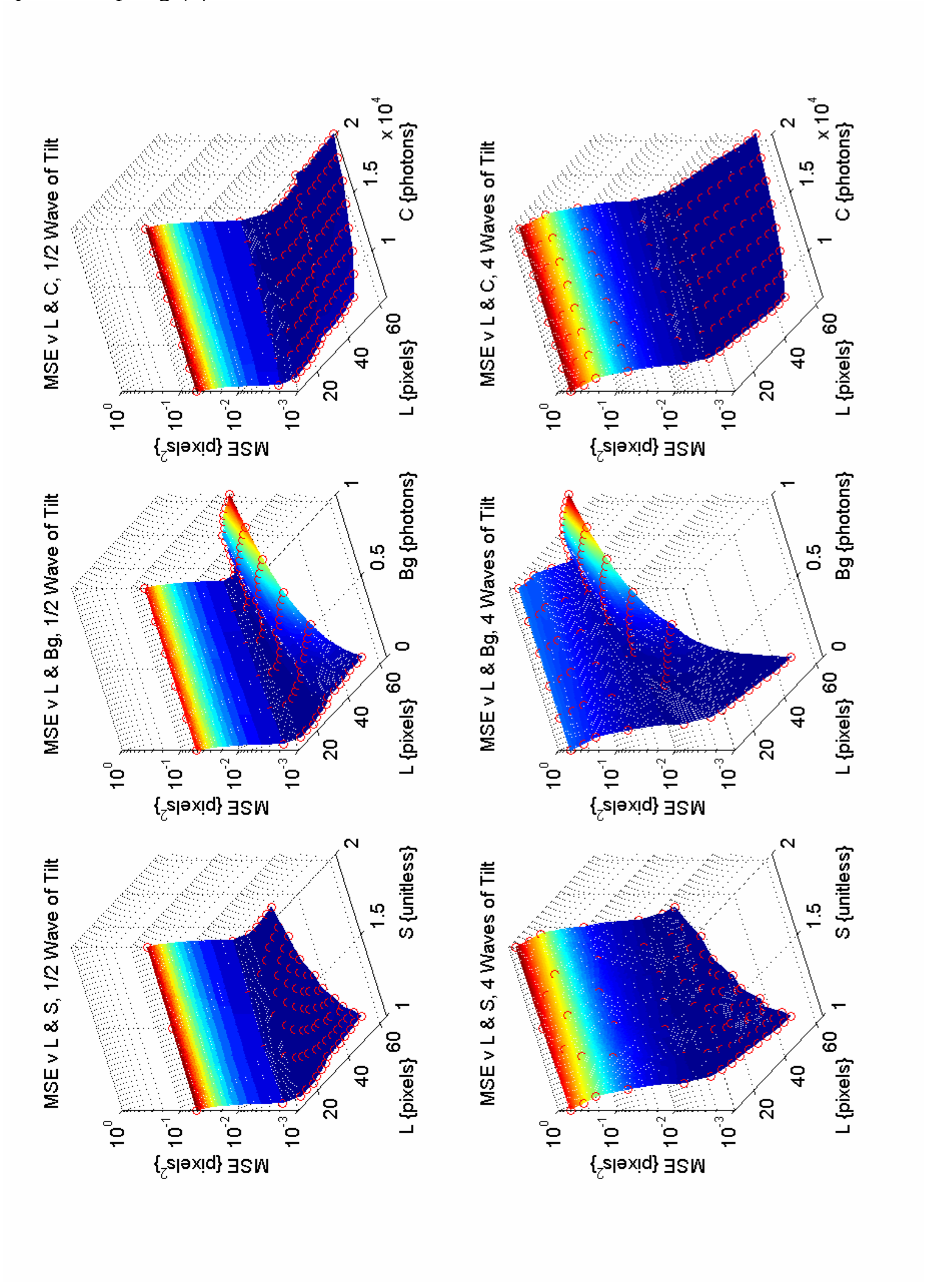
Minimum Sizes

Recommended Sizes



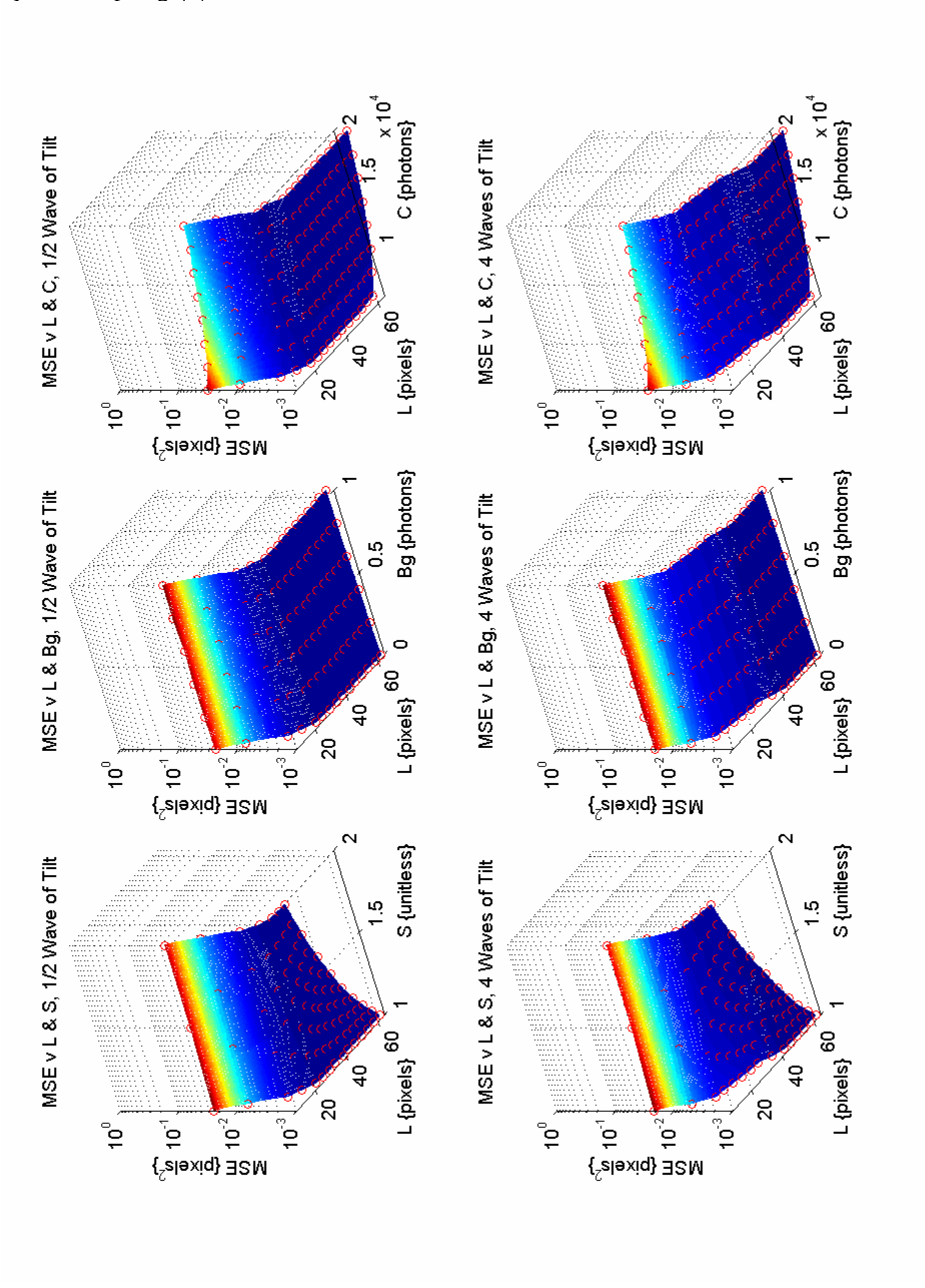
D.9. WFS Hubble swat Characterization

Parameters: Image Size (L), Intensity (C) = 8000, σ_i = 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.



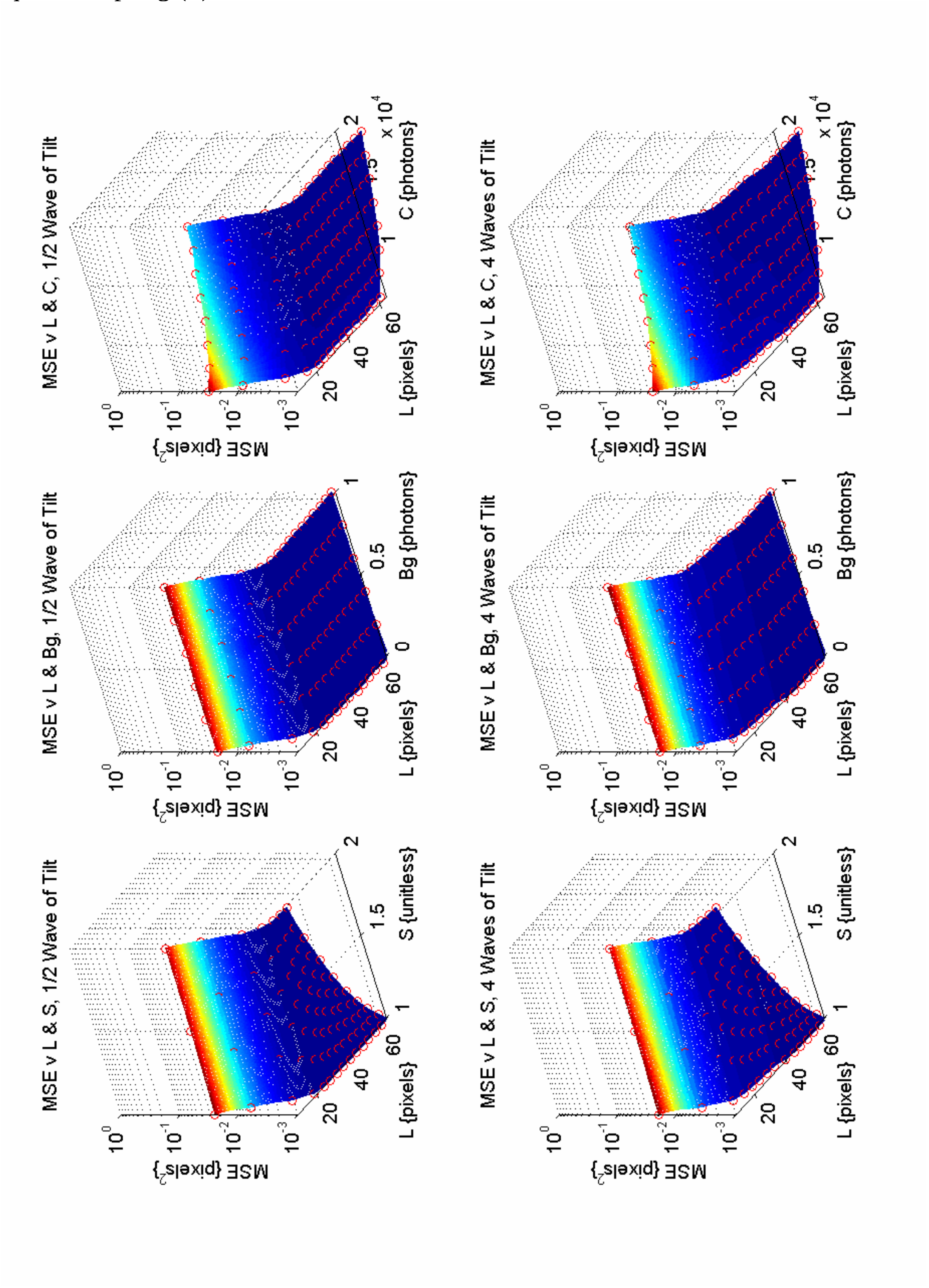
D.10. WFS Hubble mliw$_l$ Characterization

Parameters: Image Size (L), Intensity (C) = 8000, σ_i = 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.



D.11. WFS Hubble mliw$_c$ Characterization

Parameters: Image Size (L), Intensity (C) = 8000, σ_i = 2, Background (Bg) = 0, & Nyquist Sampling (S) = 1, Unless Otherwise Noted.



Bibliography

1. Baletic, James W. "A new CCD designed for curvature wavefront sensing," *Optical Detectors for Astronomy II: State –of-the-Art at the Turn of the Millenium*, 4: 283 (1999).

2. Barclay, H. T. "The SWAT wavefront sensor," *Lincoln Lab. J.* 5(1): 115 (1992).

3. Blanc, Amandine. "Marginal estimation of aberrations and image restoration by use of phase diversity," *J. Opt. Soc. Am. A,* 20(6): 1035-1045 (June 2003).

4. Cain, S. and M. M. Hayat, "Exploiting the Temporal Statistics of Atmospheric Tilt for Improved Short Exposure Imaging," *Proceedings of the 2001 Conference on Signal Recovery and Synthesis*. Albuquerque NM: November, 2001.

5. Cain, Stephen C. "Design of an image projection correlating wavefront sensor for adaptive optics," *Opt. Eng.* 43(7): 1670-1681 (July 2004).

6. Cain, Stephen C. Personal Correspondence. Jul 04 - Feb 06.

7. "Devices - Altera eStore," Online. 20 January 2006 http://www.altera.com/buy/devices/buy-devices.html

8. Goodman, J. W. *Introduction to Fourier Optics*. New York: McGraw-Hill, 1968.

9. Leon-Garcia, Alberto. *Probability and Random Processes for Electrical Engineering* (2nd Edition). Reading MA: Addison Wesley, 1994.

10. Lofdahl, Mats G. "Fast Phase Diversity Wavefront Sensing for Mirror Control," *Adaptive Optical System Technologies, Proc. SPIE* 3353 (March 1998)

11. Oppenheim, Alan V. and Ronald W. Schafer. *Discrete-Time Signal Processing* (2nd Edition). Upper Saddle River NJ: Prentice Hall, 1999.

12. Pennington, Timothy L. *Performance Comparison of Shearing Interferometer and Hartmann Wave Front Sensors*. MS Thesis, AFIT/GE/ENG/93D-31. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1993 (ADA274031).

13. Pidwirny, Michael, *Fundamentals of Physical Geography*. Online. 10 September 2005 http://www.physicalgeography.net/fundamentals/7b.html

14. Primmerman, Charles A. “Atmospheric Adaptive Optics Technology,” *Lasers and Electro-Optics Society Annual Meeting, 1992. LEOS ’92, Conference Proceeding*. 612 (16-19 November 1992)

15. “The Resolution of a Telescope – Dawes, Rayleigh and Sparrow,” *The Astroscopic Labs*, Online. Internet. 10 Nov 2004. http://www.licha.de/astro_mtf_telescope_resolution.php

16. Roggemann, Michael C. *Imaging Through Turbulence*. Boston MA: CRC Press, 1996.

17. Rousset, G. “Wavefront Sensing,” *Adaptive Optics for Astonomy*. C423 of *NATO Advanced Study Institude Series* 115-137 (1994)

18. Saleh, Bahaa E. A. and Malvin Carl Teich. *Fundamentals of Photonics*. New York: Wiley, 1991.

19. Van Trees, Harry L., *Detection, Estimation, and Modulation Theory*. New York: John Wiley & Sons, Inc., 2001.

20. Tyahla, Lori ed. “The Hubble Project - Overview,” NASA. Online. 11 Jan 2005. http://hubble.nasa.gov/overview/intro.php

21. Wiberg, D. M. “A Spatial Non-Dynamic LQG Controller: Part I, Application to Adaptive Optics,” Online. www.ucolick.org/~wiberg/ieeepart1.pdf (Submitted to IEEE for review 24 February 2004)

22. Weisstein, Eric W., “Zernike Polynomial.” From *MathWorld*—A Wolfram Web Resource. Online. 10 September 2005 http://mathworld.wolfram.com/ZernikePolynomial.html

23. Wyant, J. C. “Use of an AC Heterodyne Lateral Shear Interferometer with Real-Time Wavefront Correction Systems,” *Appl. Opt.*, 14: 2622-2626 (1975)

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 23-03-2006	**2. REPORT TYPE** **Master's Thesis**	**3. DATES COVERED** *(From – To)* Aug 2004-Mar 2006

4. TITLE AND SUBTITLE Multi-Dimensional Wave Front Sensing Algorithms for Embedded Tracking and Adaptive Optics Applications	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Wood, Christopher C., First Lieutenant, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way Bldg 640, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/06-57
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/DESA, ATTN: Dr. Victor L. Gamiz, 3550 Aberdeen Ave SE, Bldg 422, Kirtland AFB, NM 87117-5776, Comm: 505-846-4846 DSN: 246-4846 Email: Victor.Gamiz@Kirtland.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Current tracking and adaptive optics techniques cannot compensate for fast-moving extended objects, which is important for ground-based telescopes providing space situational awareness. To fill this need, a vector-projection maximum-likelihood wave-front sensing algorithm development and testing follows for this application. A derivation and simplification of the Cramer-Rao Lower Bound for wave-front sensing using a laser guide star bounds the performance of these systems and guides implementation of a vastly optimized maximum-likelihood search algorithm. A complete analysis of the bias, mean square error, and variance of the algorithm demonstrates exceptional performance of the new sensor. A proof of concept implementation shows feasibility of deployment in modern adaptive optics systems. The vector-projection maximum-likelihood sensor satisfies the need for tracking and wave-front sensing of extended objects using current adaptive optics hardware designs.

15. SUBJECT TERMS

Tracking, Adaptive Optics, Cramer-Rao Lower Bound, Maximum-Likelihood Estimation, FPGA

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			**Stephen C. Cain, PhD (ENG)**
U	**U**	**U**	UU	157	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636 ext 4281; email: stephen.cain@afit.edu

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39-18